BIBLIOTECA BASICA INFORMATICA

DIMENSION MSX



INGELEK

BIBLIOTECA BASICA INFORMATICA

DIMENSION MSX

6

INGELEK

Director editor:
Antonio M. Ferrer Abelló.

Director de producción:
Vicente Robles.

Coordinador y supervisión técnica:
Enrique Monsalve.

Colaboradores:
Angel Segado.
Patricia Mordini.
Margarita Caffaratto.
Marina Caffaratto.
Francisco Ruiz.
Jorge Juan Monsalve.
Beatriz Tercero.
Fernando Ruiz.
Casimiro Zaragoza.

Diseño:
Bravo/Lofish.

Dibujos:
José Ochoa.

ISBN del tomo: 84-85831-42-X
ISBN de la obra: 84-85831-31-4
Fotocomposición: Pérez Díaz, S. A.
Imprime: Héroes, S. A.
Depósito Legal: M-42.894-1985



INDICE

PROLOGO



n ordenador MSX, sea cual sea la marca, es un dispositivo electrónico muy sofisticado que cumple el primer estándar referente a los ordenadores domésticos.

MSX es la contracción de Microsoft EXtended; en cuanto al lenguaje supone un BASIC, realizado por Microsoft, para ordenadores que tengan como CPU el Z80. La elección de este microprocesador de 8 bits, sin lugar a dudas un poco anticuado, tiene su justificación: siempre que situemos al ordenador MSX en su ámbito doméstico, en el cual la facilidad de uso y la gran disponibilidad de software (no sólo de juegos) y de periféricos son factores más importantes que elegir una solución más poderosa, sofisticada y rápida, como podría ser cualquiera basada en microprocesadores de 16 ó 32 bits.

Para que se pueda hablar con fundamento de estándar, la estructura interna del procesador no debe tener secretos para las casas de software que quieran desarrollar programas, y la tecnología de los aparatos tiene que ser de fácil acceso para los diseñadores del hardware auxiliar. La indiscutible fiabilidad de la CPU Z80, comprobada durante años en una gran cantidad de aplicaciones, su fácil manejo y el patrimonio de las muchas experiencias que se han basado en ella a lo largo del tiempo, la convierten en el elemento ideal para lograr los objetivos mencionados.

En resumen: todos los ordenadores MSX tienen una compatibilidad total de programas y periféricos; su BASIC, el mismo para todos, es rico, poderoso y fácil de usar, lo que contribuye a acentuar el aspecto "friendly" (amigable) de estos ordenadores.

"Dimensión MSX" quiere ser una introducción completa, aun-

que no exhaustiva, a estos nuevos, poderosos y versátiles ordenadores. Sin pretender sustituir al manual de referencia (aunque, en honor a la verdad, la calidad de dichos manuales es muy poco homogénea dentro del estándar) queremos ofrecerle una visión global de las prestaciones que ofrecen los sistemas MSX: encontrará las instrucciones para conectar su ordenador y poder hacer que funcione, un repaso rápido (pero no superficial) de las instrucciones del BASIC MSX, un estudio de cuatro apartados en los que el estándar MSX ha alcanzado los máximos niveles cualitativos: gráficos, sonido, ficheros e interrupciones; para terminar, una serie de subrutinas, que podrá usar en sus programas, explicadas a fondo, instrucción por instrucción, junto con consejos sobre cómo mejorar su estilo de programación o hacer más eficaces sus programas.

CAPITULO I

LOS ORDENADORES MSX

Acercándonos al ordenador

El ordenador es una máquina capaz de resolver problemas de muy variada naturaleza, una vez que se le enseña cómo hacerlo; esta idea puede que no sea tan evidente para el que se acerca por primera vez a estas máquinas, así que la estudiaremos más detenidamente.

El ordenador puede hacer muchas cosas, pero no inventa nada. Es un perfecto ejecutor, mucho más rápido, fiable y, sin duda, obediente que el hombre, pero no toma iniciativas. Todo lo que puede hacer es llevar a cabo las instrucciones que se le den, por muy complejas y numerosas que sean.

En base a algunas de estas instrucciones puede "hacer elecciones", pero aun en este caso no tiene ninguna autonomía: elige según los criterios marcados por la persona que lo programa. Esto obliga a que los criterios tengan que ser claros, sin equívocos ni ambigüedades, y rigurosos.

El conjunto de operaciones que se llevan a cabo para permitir que el ordenador pueda resolver un determinado problema es lo que se llama "programación". La justificación de esta palabra no es difícil de entender: el ordenador no se limita a ejecutar las instrucciones una por una, según se le vayan dando, sino que es capaz de recordar cierto número de ellas y ejecutarlas, según los deseos del usuario, en una secuencia bien definida y con un orden preestablecido; ese conjunto de órdenes codificadas representa entonces un verdadero "programa" de trabajo.

Las instrucciones para un ordenador tienen que ser expresa-

das en un lenguaje especial que éste pueda entender. Para aclarar el concepto vamos a recurrir a una analogía muy simple, referente a una máquina que probablemente nos es más familiar: cuando pulsamos el interruptor de una lavadora para dar comienzo al lavado en realidad lo que hacemos es comunicar un mensaje a la lavadora ("ponte en marcha") en su lenguaje, de forma que nos pueda entender y, efectivamente, inicie el lavado. La situación, en el caso del ordenador, no es tan sencilla, como corresponde esperar de la máquina más compleja que es. Existen varios lenguajes, a distintos "niveles", por medio de los cuales podemos comunicarnos con el ordenador. Si nos imaginamos una serie de planos a diversos niveles se dice que en el nivel más bajo está el "código máquina", constituido por dos únicos símbolos: el Ø y el 1, que es el utilizado por la mayoría de los dispositivos electrónicos de los que se compone el ordenador; es el más lejano al hombre y éste casi nunca lo utiliza directamente (indirectamente sí, pues, en definitiva, cualquier otro lenguaje deberá ser traducido al "máquina" para que el ordenador pueda entenderlo). En el nivel más alto está el lenguaje del hombre, también llamado "lenguaje natural", basado, como sabemos, en un alfabeto de bastantes letras y rico en reglas sintácticas y gramaticales que lo hacen bastante difícil de aprender.

Entre estos dos extremos hay toda una serie de lenguajes situados en niveles intermedios de proximidad al hombre, o a la máquina. En la figura 1 están representadas las categorías de lenguajes que se encuentran en los distintos niveles; en un nivel inmediatamente superior al código máquina se encuentran los *lenguajes* ensambladores (assembler o assembly), que utilizan códigos simbólicos en lugar de los Ø y 1, pero que tienen una estructura similar a la del código máquina; en el siguiente nivel encontramos los *lenguajes evolucionados,* que utilizan "frases" muy parecidas a las del lenguaje natural (inglés, claro está), pero con reglas mucho más rígidas.

En cualquier ordenador, una de las cosas que destaca es el teclado, muy parecido al de una máquina de escribir, aunque con algunas teclas adicionales que aprenderá a utilizar muy rápidamente. Las instrucciones para resolver un determinado problema, los datos que habrá que manejar y cualquier otra orden que desee dar al ordenador se comunican a través del teclado. Generalmente, todo lo que introduzcamos por él aparecerá visualizado simultáneamente en la pantalla, así como los resultados de la ejecución y cualquier otro mensaje que el ordenador considere que debe comunicarnos, por ejemplo, los eventuales mensajes de error, con los que la máquina nos informa que algo en las instrucciones no es correcto o no está claro. Cuando creamos necesario conservar la información que aparece en la pantalla de forma de-

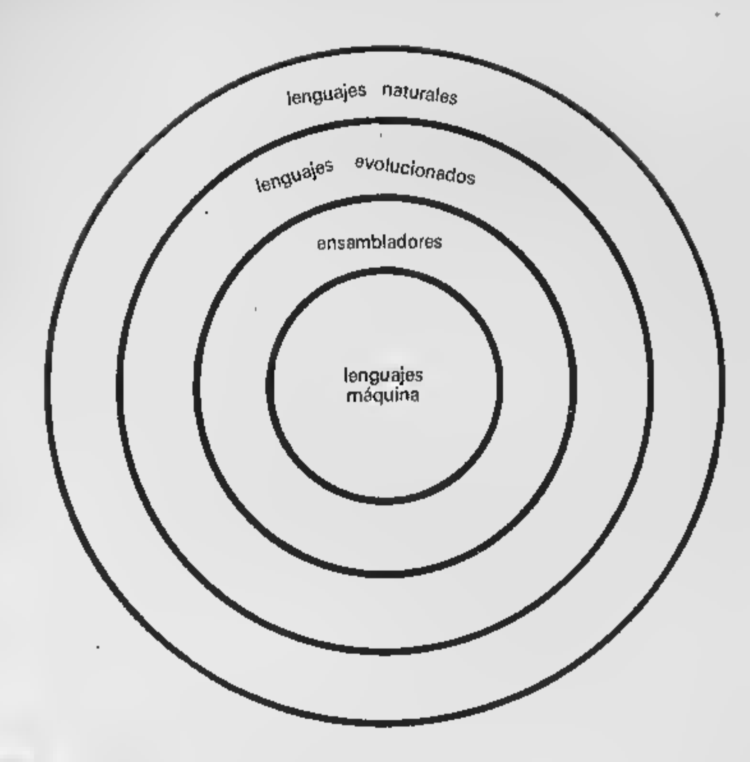


Fig. 1 - Lenguajes de comunicación entre hombre y máquina.

finitiva, podemos dar la orden de imprimirla sobre papel, tarea de la que se ocupa la impresora.

El teclado forma parte de los llamados dispositivos de entrada (input), mientras que la pantalla y la impresora pertenecen al grupo de los de salida (output). En general, todos ellos se conocen con el nombre de dispositivos o periféricos de entrada/salida (E/S o bien I/O). Las instrucciones y datos que introduzcamos mediante el teclado se conservarán en una parte del ordenador que se conoce como memoria central, donde toda la información se representa en forma de ceros y unos. En la figura 2 puede ver una configuración típica.

Cómo conectarlo

Estará, sin duda, impaciente por olvidarse de tanta palabrería y poder conectar su nuevo ordenador para que por fin funcione,

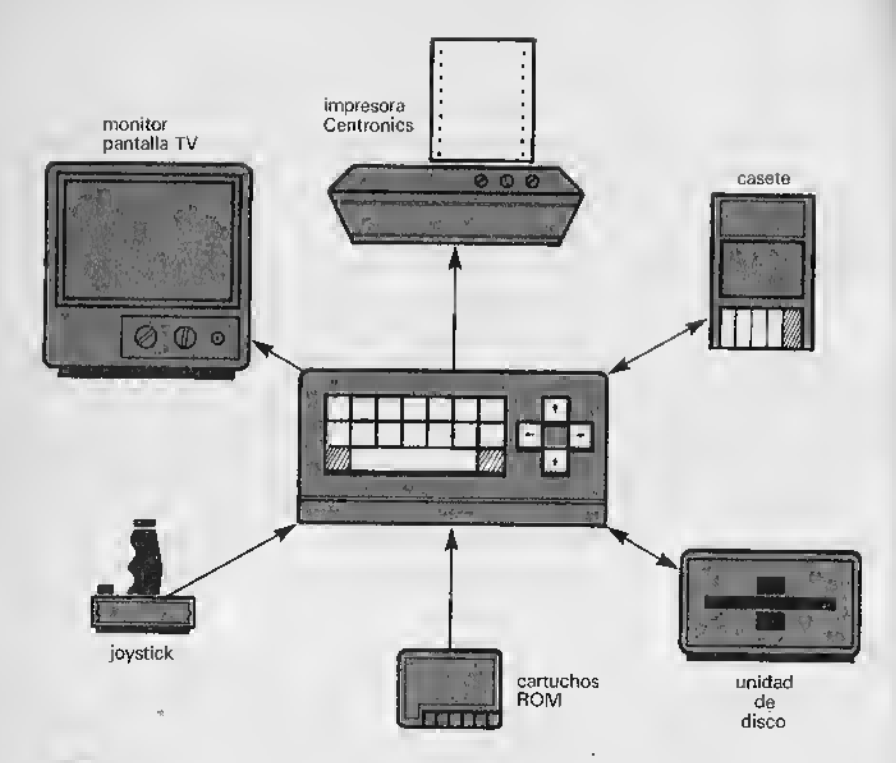


Fig. 2 · Sistema MSX con sus diversas opciones.

pero ¡cuidado!, la prisa es mala consejera, sobre todo cuando se trata máquinas tan sofisticadas como su ordenador. Procedamos con calma y orden.

Saque el ordenador del embalaje con cuidado y compruebe que estén:

- los cables de alimentación. Algunos MSX, como el SONY y el TOSHIBA tienen el alimentador incorporado; otros, como el PHILIPS, tienen un alimentador externo;
- el cable de conexión a la grabadora. Lo reconocerá en seguida porque tiene en un extremo una clavija DIN, del tipo utilizado en los aparatos de sonido, y en el otro, tres clavijas de color blanco, negro y rojo (Fig. 3);
- el cable de conexión a la pantalla o monitor, si los hubiere;
- el manual del usuario.

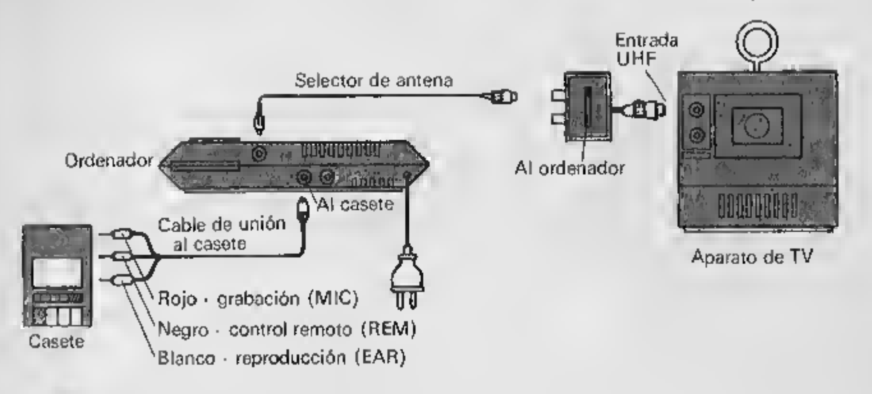


Fig. 3 - Conexión del ordenador a la pantalla de TV y al casete.

Antes de enchufar su ordenador examínelo detenidamente: algo tan sofisticado merece cuidados especiales. Deberá familiarizarse con la disposición de los numerosos conectores que hacen de su MSX un sistema expandible cuyas posibilidades pueden crecer junto con sus exigencias. Gracias a las posibilidades de conexión que ofrece, el sistema MSX es realmente versátil: puede convertirse en una pantalla para videojuegos, un sistema de gestión para una empresa pequeña, un sistema de escritura electrónica o un instrumento musical.

Debería encontrar fácilmente (si no es así, busque ayuda en el "manual del usuario" del ordenador) el interruptor general, los conectores para los joysticks (generalmente hay dos, idénticos, identificados por las letras A y B o por los números 1 y 2), el de la impresora (en algunas versiones, por ejemplo en el PHILIPS, no existe, ya que utilizan un slot), dos slots (ranuras) para conectar cartuchos, unidades de disco, etc. (en algunas versiones, un slot se encuentra en el plano del teclado y otro en la parte posterior del ordenador, mientras que en otras ambos se localizan en el plano del teclado), el conector para la grabadora y el de la televisión o monitor.

El ordenador puede trabajar perfectamente por sí solo, pero para que pueda comunicarse con usted es imprescindible lo que en términos técnicos se llama dispositivos de salida (output). En el caso del estándar MSX se puede utilizar como tal un televisor (en blanco y negro o en color) o, mejor aún, un monitor. La ventaja del monitor es que las imágenes son mucho más nítidas (el inconveniente, lógicamente, es que no se puede utilizar para ver programas televisivos).

Si su televisor en blanco y negro es de un modelo antiguo puede que tenga un conector de antena un poco distinto al que viene con el ordenador. Los televisores de hace algunos años utilizaban un cable plano con dos hilos para conectarse a la cajita que lleva la señal desde la antena al televisor (demodulador). En este caso (actualmente muy raro) tendrá que pedir un adaptador de 300 a 75 ohmios en una tienda de electrodomésticos o de electrónica, o dirigirse a la tienda donde haya comprado el ordenador; allí, sin duda, le podrán ayudar.

Otro dispositivo muy útil, indispensable para el que desee conservar sus programas, es la grabadora o magnetófono a casete. El ordenador dispone de una memoria interna en la que graba los programas y los datos que le proporcionamos a través del teclado; esta memoria, llamada RAM (Random Access Memory, memoria de acceso aleatorio), es muy rápida, pero, desafortunadamente, pierde todo su contenido (se "olvida" de él) cuando apagamos el ordenador. Para poder conservar de forma permanente datos y programas tenemos dos posibilidades, utilizar un cartucho de memoria no volátil (data cartridge) o una grabadora (no consideramos por ahora la opción de las unidades de disco). Los "data cartridge" son memorias RAM con una batería incorporada que las alimenta constantemente. Se tienen que conectar a los slots para cartuchos y hay que transferirles el contenido de la memoria. Su inconveniente es la pequeña capacidad y el costo un poco excesivo; en cambio, son muy rápidos.

Memoria interna y externa

Las memorias para ordenadores se distinguen en volátiles y no volátiles. Las primeras pierden su contenido cuando se dejan de alimentar, mientras que las segundas lo conservan de forma permanente. Entre las primeras está la RAM, muy rápida, se encuentra dentro de su ordenador en grandes cantidades. Tienen el aspecto de cajitas negras con muchas patillas (por desgracia, la mayor parte de los componentes son también así).

Entre las no volátiles está la ROM (Read Only Memory, memoria de sólo lectura) de aspecto parecido a la RAM. Dentro de su ordenador son las encargadas de mantener almacenado el BASIC MSX, listo para su uso desde que se enciende el ordenador. Otras memorias no volátiles son las de tipo magnético, como las cintas o los discos. El principio de grabación es el mismo para ambas: consiste en la magnetización de óxidos de hierro depositados sobre un soporte de plástico. Lo realmente distinto, además del soporte físico, es la forma de grabar los datos: la cinta magnética es un dispositivo secuencial, es decir, la información se graba

una tras otra y para localizar un dato hay que examinar todos los que lo preceden, nos interesen o no, con la consiguiente pérdida de tiempo. El disco, en cambio, es un dispositivo de acceso directo, pues permite acceder a la información que se desea directamente, sin tener que analizar las demás. Prácticamente, la diferencia se manifiesta en el tiempo de acceso a los datos: con una cinta se necesitan algunos minutos para grabar un programa de tamaño medio y muchos más para volverlo a cargar en el ordenador; con el disco bastan pocos segundos, tanto para grabarlo como para buscarlo y cargarlo.

Es obvio que el precio refleja las prestaciones: las unidades de disco, que hacen uso de sistemas mecánicos muy refinados y precisos, son cerca de diez veces más caras que las grabadoras (que, además, no suelen tener que comprarse, pues sirve incluso la que usamos para escuchar o grabar música).

El método de grabación más accesible para el que empieza a usar un ordenador es el casete. Se puede tratar de los casetes normales de audio que utilizamos para grabar música; gracias a ellos el ordenador grabará en las cintas magnéticas (las populares "casetes") una serie de sonidos que luego puede leer e interpretar. No es aconsejable utilizar cintas de muy larga duración, ya que dañan el motor de arrastre de la grabadora, además de hacer que la búsqueda sea larga y aburrida. Lo mejor es usar casetes C1Ø o C2Ø (cuya duración es de 1Ø y 2Ø minutos, respectivamente), que se pueden encontrar en los comercios de ordenadores o en algunas tiendas de electrodomésticos.

Por lo que se refiere a la grabadora, los ordenadores MSX permiten la utilización de cualquier tipo (de buena calidad, eso sí) con excelentes resultados. Para garantizar una fiabilidad mayor, algunos fabricantes de ordenadores han decidido fabricar grabadoras especiales, diseñadas para su uso informático específicamente.

Conectar ordenador y grabadora es una operación muy sencilla, pero hay que tener mucho cuidado para no cometer errores. Es preciso que a lo largo de esta operación el ordenador y la grabadora estén apagados, para evitar posibles daños al ordenador. El conector DIN del cable se tiene que enchufar en la parte posterior del ordenador, en el lugar indicado con la palabra RECORDER, mientras que los tres jacks (clavijas) se tienen que conectar a la grabadora de la forma siguiente:

- jack negro: sirve para el control remoto del motor de la grabadora por parte del ordenador y hay que enchufarla en la salida REM de aquélla. Es más pequeña que las otras dos, lo que hace imposible; su confusión. Algunas grabadoras no tienen esta toma, esto no impide su utilización con el ordenador: es suficiente con dejar la clavija desconectada. El in-



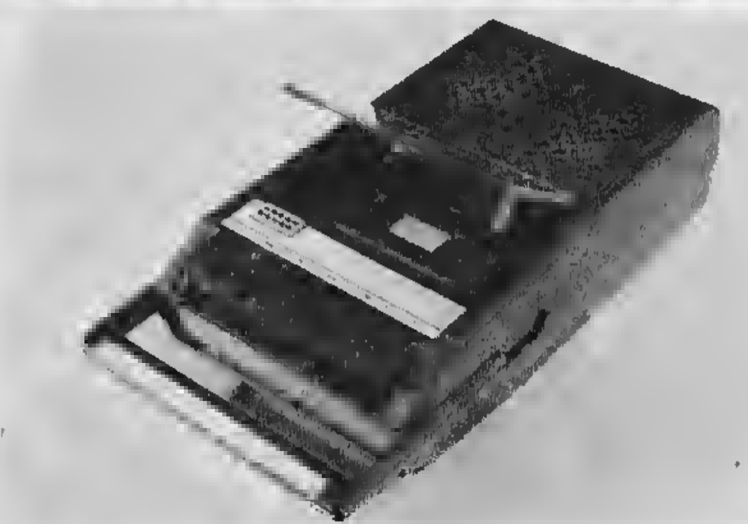


Fig. 4 · Un casete de audio se puede emplear como memoria externa de bajo coste.

conveniente consistirá en que tendremos que arrancar y parar manualmente la cinta, en lugar de dejar que el ordenador se encargue de todas las operaciones;
jack rojo: permite transferir datos del ordenador a la grabadora en la fase de salvar programas o datos; hay que conectarlo a la entrada MIC (micrófono);

- jack blanco (o negro, pero con cable blanco): permite transferir datos desde la grabadora al ordenador en la fase de lectura de datos o de programas. Hay que conectarlo a la salida EAR (auricular).

Si se confunde e invierte las clavijas roja y blanca no podrá utilizar la grabadora, pues el ordenador intentará enviar señales por el canal que debe utilizar para leerlas, y viceversa. Si se produjera un mal funcionamiento de la grabadora, compruebe primero dicha conexión.

Las unidades de disco, evidentemente más caras, permiten, sin embargo, transformar un ordenador doméstico en un sistema capaz de manejar grandes cantidades de datos. El sistema operativo MSX recoge cosas buenas de otros dos muy conocidos: el CP/M y el MS-DOS. Desafortunadamente, sobre algunos temas no se ha logrado un acuerdo total entre los varios fabricantes; así, por ejemplo, algunos utilizan discos de 5"1/4 y otros de 3"1/2. Se mantiene de todas formas la intercambiabilidad de las unidades de disco entre ordenadores MSX, lo que hace posible utilizar, por ejemplo, la unidad para microfloppys de Sony con un Philips.

Otros periféricos característicos son los joystick (Fig. 5). Permiten, conjuntamente con los numerosísimos programas disponibles, transformar su ordenador MSX en una máquina de videojuegos de muy alto nivel. La conexión de los joysticks es... un juego: es suficiente enchufarlos en las tomas correspondientes. El conector no es simétrico, sino que tiene forma de D, con un lado más ancho que el otro, por lo que no tiene que pensar cuál es la posición correcta: ¡sólo hay una! Cuidado si no entra: no lo presione, sólo gírelo. Los conectores para joysticks son dos; si quiere jugar sólo con uno tiene que conectarlo a la toma identificada con el número 1 o con la letra A.

Para utilizar el ordenador como una potente máquina de escribir electrónica, o simplemente para disponer de listados o resultados impresos, es indispensable una impresora. También para estos dispositivos la compatibilidad es total; los modelos disponibles van desde impresoras de 80 columnas, gráficas, con matrices de puntos (Philips), a los pequeños plotters, que dibujan sobre hojas de formato A4 y que se pueden usar como impresoras (Sony).



Fig. 5 - Joystick de equipos MSX.

El teclado

Los distintos modelos de ordenadores MSX (Fig. 6) tienen teclados con ligeras diferencias, pero sólo en su apariencia; las teclas y las funciones que desarrollan son siempre las mismas. En este párrafo describiremos su uso sin hacer referencia a ningùn modelo en particular; la figura 6 y los manuales de su ordenador le ayudarán a identificar las teclas según se vayan describiendo. Si conoce cómo es el teclado de una máquina de escribir observará inmediatamente que el de su MSX es similar, aunque tiene algunas diferencias. Las letras tienen la disposición del estándar anglosajón: leyendo las de la primera fila de izquierda a derecha forman la palabra QWERTY, y no QZERTY, como corresponde al estándar europeo (ciertamente poco difundido). Ciertas teclas presentes en el teclado MSX no existen en las máquinas de escribir y, de hecho, tienen un significado especial.

Le aconsejamos que vaya pulsando y comprobando el funcionamiento de las distintas teclas según expliquemos su uso para ir tomando confianza con el teclado. No tema estropear el orde-



Fig. 6 - Típico teclado MSX.

nador: nada de lo que puede hacer usted desde el teclado lo puede dañar (¡a no ser que use un martillo!). Para conseguir que el ordenador se olvide de cualquier tontería que escribamos, basta apagarlo y volverlo a encender: dócilmente estará otra vez listo para ejecutar nuestras órdenes. Los comandos descritos a continuación y que no corresponden al nombre de ninguna tecla se tienen que escribir letra por letra (como con una máquina de escribir) y, al final, pulsar la tecla RETURN.

SHIFT

Al igual que en una máquina de escribir, podemos obtener las letras mayùsculas pulsando una tecla; en este caso la que tiene escrito SHIFT. Además nos permite obtener el símbolo que se encuentra en la parte superior de algunas teclas: para obtener el símbolo %, por ejemplo, bastaría pulsar las teclas SHIFT y 5 simultáneamente.

CAPS

Para referirse de forma continua a las letras mayúsculas hay que pulsar CAPS, abreviatura de CAPITALS (mayúsculas en inglés); generalmente, cuando pulsamos esta tecla se enciende un piloto para recordarnos que estamos escribiendo todo en mayúsculas. Su misión se acaba aquí, por lo que para obtener los símbolos superiores de las teclas hay que recurrir siempre a SHIFT.

GRAPH y CODE

Además de SHIFT existen otras dos teclas que permiten obtener caracteres especiales cuando se pulsan conjuntamente con otras: se trata de GRAPH y CODE. La primera permite representar los caracteres gráficos correspondientes a la de la parte inferior de la tecla (Fig. 7); en unión con SHIFT produce los de la parte superior.

CODE y CODE+SHIFT permiten obtener conjuntos ("sets") de caracteres alternativos, como pueden ser las letras griegas o vocales acentuadas de distintos idiomas.

RETURN

RETURN es una tecla presente en todos los ordenadores con la misma función, aunque en algunos se llama ENTER. Sirve para indicar al ordenador que hemos acabado de escribir una línea u orden. Mientras pulsemos teclas sin dar RETURN, el ordenador se limitará a visualizar los caracteres correspondientes en la pantalla, sin tomarlos en consideración; en cuanto pulsamos RETURN los examina detenidamente y comprueba si se trata de una línea



Fig. 7 - Otro teclado MSX.

de programa, que hay que memorizar para su posterior ejecución, o de un comando que hay que ejecutar en el acto. Recuerde que hay que terminar cada línea de programa pulsando RETURN.

Control del cursor (↑,↓,→,←)

Las teclas de control del cursor se encuentran generalmente en la parte inferior derecha, separadas de las demás. El cursor es el cuadradito que en la pantalla indica dónde aparecerá el próximo carácter que tecleemos. Para moverlo basta pulsar una cualquiera de estas cuatro teclas, lo que provocará su desplazamiento en la dirección y sentido que marca la flecha impresa en ellas.

TAB HOME CLR

Para obtener un movimiento rápido del cursor existe la tecla TAB, que desplaza el cursor ocho caracteres en la misma línea.

Para llevar el cursor a la esquina superior izquierda (posición inicial o "home") directamente es suficiente pulsar la tecla HOME.

Si quiere borrar la pantalla, pulse HOME junto con SHIFT: obtendrá la función CLR (abreviación de CLEAR, "borra" que elimina todos los caracteres de la pantalla y lleva el cursor a la esquina superior izquierda. El mismo resultado se puede conseguir tecleando CLS seguido por RETURN (se trata de una función del BASIC que veremos más adelante).

STOP

Otra tecla muy importante es la que lleva la inscripción STOP. Si la pulsamos durante la ejecución de un programa, la ejecución de éste se bloqueará hasta que no la apretemos de nuevo. Si es accionada junto a CTRL detiene definitivamente la ejecución; para volver a empezar será necesario enviar el comando CONT.

CTRL

CTRL es una tecla extraña que por sí misma no tiene ningún efecto, pero que, en unión de otras, es muy poderosa. Acabamos de ver lo que hace junto a STOP; la tabla siguiente nos ofrece un cuadro más general.

TECLAS DE FUNCION

En la parte superior izquierda del teclado hay cinco teclas, identificadas con F1(F6)... F5(F10). Son las llamadas teclas de función; cada una está asociada a un comando, de forma que el or-





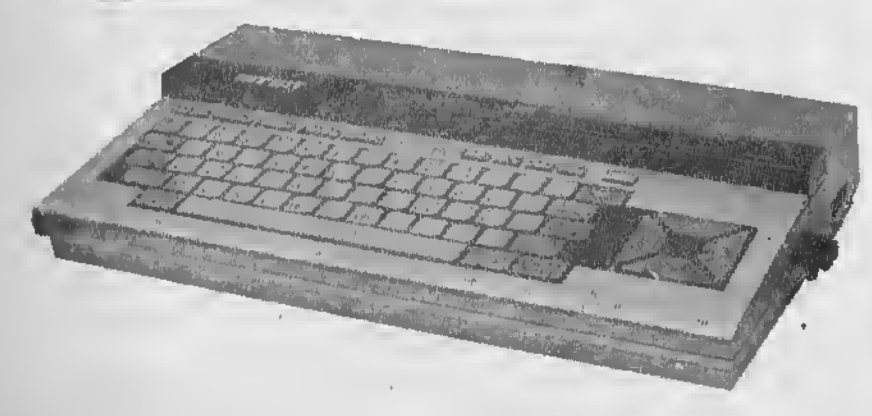


Fig. 8 · Disposición y forma diversa de las teclas de control del cursor en ordenadores MSX.

FUNCIONES DE LA TECLA CTRL

Código ASCII	Tecla (+ CTRL)	Tecla equivalente	Función
1	A		Sets de caracteres alternativos
2	B		Cursor a la palabra anterior
3	C		Fin numeración automática
4	D		Ninguna
5	E		Borra a partir del cursor
6	F		Cursor a la palabra siguiente
7	G		Señal acústica
8	H	BS	Borra carácter precedente
9	I	TAB	Traslada el cursor 8 posiciones a la izquierda
10	J		Cursor a la línea siguiente
11	K	HOME	Cursor a la esquina superior izquierda
12	L	CLR	Borra la pantalla
13	M	RETURN	Baja el cursor a la línea siguiente y envía al ordenador los caracteres que hubiera en la anterior
14	N		Cursor al final de línea
15	O		Ninguna
16	P		Ninguna
17	Q		Ninguna
18	R	INS	Inserta carácter desplazando los demás a la derecha
19	S		Ninguna
20	T		Ninguna
21	U		Borra línea
22	V		Ninguna
23	W		Ninguna
24	X	SELECT	Definida por programa
25	Y		Ninguna
26	Z		Ninguna
27	[	ESC	Definida por programa
28	\	→	Cursor a la derecha
29	]	←	Cursor a la izquierda
30		↑	Cursor arriba
31	.	↓	Cursor abajo


Tabla 1 - Funciones de la tecla CTRL.

denador lo escribe por nosotros en cuanto se pulsa la tecla correspondiente. Por ejemplo, en lugar de teclear el comando LIST letra por letra es suficiente pulsar F4 y RETURN para obtener el mismo efecto. La relación completa de los comandos asociados a cada tecla de función se encuentra en la tabla siguiente. Según vayamos aprendiendo el uso de las instrucciones del MSX BASIC encontraremos cada vez más útiles las teclas de función. Por otro lado, podemos asociarlas también cualquier otra cadena de caracteres que nos interese con la instrucción KEY.

RESET

La función de esta tecla es muy peligrosa. En algunos ordenadores MSX, como el TOSHIBA, ni siquiera existe; en otros está

TECLAS DE FUNCION		
Tecla	**Comando**	**Descripción**
F1	color	Definición del color
F2	auto	Numeración automática
F3	goto	Instrucción GOTO
F4	list	Lista el programa en memoria
F5	run	Ejecuta el programa en memoria
F6	color 15,4,7	Valores por defecto del color
F7	cload"	Carga un programa desde casete
F8	cont	Continúa la ejecución interrumpida
F9	list.	Presenta la última línea ejecutada
F10	cir + run	Borra pantalla y ejecuta programa en memoria


Tabla 2 - Teclas de función.

muy bien escondida (en el PHILIPS se encuentra en la parte posterior) y en algunos más, como el SONY, está en una posición visible pero protegida por un marco que impide una pulsación accidental. En cualquier caso, pulsar RESET equivale a apagar y volver a encender el ordenador: la memoria se borra y el eventual programa presente se pierde. Atención entonces a no destruir accidentalmente en una fracción de segundo el trabajo de horas.

Edición de programas

Existen otras tres teclas que permiten modificar fácilmente los caracteres que aparecen en la pantalla, ya pertenezcan a programas en BASIC o a poesías de Machado (lo primero es, sin duda, más probable). Se trata de las teclas de "edición": BS, DEL, INS.

Supongamos que usted escriba las dos líneas siguientes de un programa (si tiene curiosidad por descubrir el significado de cada instrucción, un poco de calma: en los próximos capítulos se lo aclararemos):

```
10 PRINT "Hola, soyy tu"
20 PRINT "ordendor MSX"
```

Copie exactamente lo que está escrito (¡incluso los errores!) y acuérdese de pulsar la tecla RETURN al final de cada línea. Teclee luego la palabra RUN seguida por RETURN. En la pantalla aparecerá:

```
Hola, soyy tu
ordendor MSX
```


lo cual no es muy elegante. Para corregir este tipo de errores puede seguir el método, eficaz pero brutal, de volver a escribir toda la línea equivocada (con la pérdida de tiempo y el cansancio que implica), o aprender a utilizar las teclas de edición. Veamos cómo.

Teclee LIST seguido por RETURN (ya no especificaremos más que a cualquier comando o línea le tiene que seguir RETURN, excepto en casos muy particulares; muchas de las veces que parece que el ordenador no está haciendo nada, es que nos hemos olvidado de pulsar RETURN).

En contestación a su LIST aparecerá en la pantalla:

```
10 PRINT "Hola, soyy tu"
20 PRINT "ordendor MSX"
```

y luego Ok, con lo que el ordenador le comunica que está listo para cumplir sus siguientes órdenes.

BS

Pulse la tecla con la flecha hacia arriba hasta posicionar el cursor en la línea 10 y luego la de la flecha hacia la derecha hasta situarlo a la izquierda de "tu" y pulse BS. Como por arte de magia la "y" desaparece y todos los caracteres a su derecha se mueven una posición para ocupar el hueco que ha quedado libre. La línea aparece ahora como:

```
10 PRINT "Hola, soy tu"
```

DEL

Hubiera podido obtener el mismo resultado llevando el cursor sobre la primera "y" para pulsar DEL después. La diferencia está en que DEL borra el carácter situado a la derecha del cursor, mientras que BS hace lo propio con el de la izquierda.

Si una vez realizada cualquiera de estás operaciones le satisface el resultado, pulse RETURN: la nueva línea sustituirá a la antigua. Para cerciorarse puede dar otra vez el comando LIST, que le presentará la situación siguiente:

```
10 PRINT "Hola, soy tu"
20 PRINT "ordendor MSX"
```

INS

Para corregir también la línea 20 mueva el cursor hasta la letra "d" y pulse INS: el cursor cambiará de aspecto, haciéndose más

bajo, y cualquier carácter que teclee a partir de ahora colocará detrás de la "n" y antes de la "d", desplazando a todos los situados a la derecha una posición. Para salir del modo inserción basta usar una cualquiera de las teclas de control lateral del cursor (no las de cambio de línea, pues la modificación no sería memorizada). Verifique otra vez con LIST que la corrección ha sido grabada y ejecute el programa con RUN. La frase

```
Hola, soy tu
ordenador MSX
```

recompensará sus esfuerzos.

Las líneas se suelen numerar de 10 en 10 para permitir la inserción de nuevas instrucciones con números de línea intermedios. Si, por ejemplo, introduce la línea:

```
15 PRINT "maravilloso"
```

Con el comando LIST observará que el programa ha sido modificado de esta forma:

```
10 PRINT "Hola, soy tu"
15 PRINT "maravilloso"
20 PRINT "ordenador MSX"
```

Observe cómo el ordenador se ha encargado de ordenar las líneas en base a su número. Para borrar una línea es suficiente teclear su número y dar RETURN:

```
15
```

elimina la línea que acabamos de introducir.

Si quiere borrar de la memoria del ordenador los programas que estén en ese momento en ella, tiene que dar el comando NEW (cuidado con no perder así por equivocación programas que hayan costado horas de trabajo, ya que no hay forma de recuperarlos).

Teclee ahora NEW e introduzca este nuevo programa:

```
10 PRINT "INSTRUCCION 1"
20 PRINT "INSTRUCCION 2"
30 PRINT "INSTRUCCION 3"
40 PRINT "INSTRUCCION 4"
50 PRINT "INSTRUCCION 5"
60 PRINT "INSTRUCCION 6"
```


que visualiza en la pantalla al ejecutarse:

```
INSTRUCCION 1
INSTRUCCION 2
INSTRUCCION 3
INSTRUCCION 4
INSTRUCCION 5
INSTRUCCION 6
```

Para borrar más de una línea consecutiva de programa a la vez es mejor que utilice la instrucción DELETE seguida por los números de la primera y la última línea que quiera eliminar, separados por un guión. Eliminemos las líneas número 30 al 50:

```
DELETE 30-50
```

Listando el programa lo verá así:

```
10 PRINT "INSTRUCCION 1"
20 PRINT "INSTRUCCION 2"
60 PRINT "INSTRUCCION 6"
```

Eliminar ahora el "agujero" producido en la numeración de las líneas puede conseguirse con el comando RENUM, que renumera las líneas. Si no especificamos ningún valor después del comando, la renumeración se efectúa a partir del primer número de línea con incrementos de 10 (10, 20, 30...), pero podemos especificar, si queremos, el nuevo número de la línea inicial, el antiguo y el incremento entre una línea y la siguiente.

Otra prestación muy útil de los MSX a la hora de escribir programas largos es el sistema de numeración automática de la línea, que se activa con el comando AUTO. Podemos declarar dos números después del comando: el primero indica el número de línea inicial y el segundo el incremento. Siempre que termine una línea (con RETURN) el ordenador hará aparecer el número de la línea siguiente. Si dicha línea ya existe aparecerá un asterisco al lado del número; si pulsamos RETURN se mantendrá la línea antigua, pero de otra forma la nueva sustituirá a la anterior. Para poner fin a la numeración automática hay que pulsar CTRL-STOP o CTRL-C, es decir, simultáneamente la tecla CTRL y la de STOP o la de CTRL y la C.

Realización y conservación de un programa

Para escribir, corregir y ejecutar cualquier programa tiene que seguir el procedimiento siguiente:

1. Teclee NEW (limpie la memoria de posibles programas anteriores).

2. Introduzca todo el programa. Utilice, si quiere, la opción AUTO, que numera automáticamente las líneas, y ponga mucha atención en no cometer errores de escritura. Por lo que a esto último se refiere, son importantes algunos consejos para el principiante:

a) No confunda el número Ø con la letra O.
b) Cada línea de programa empieza con un número de línea y termina con RETURN.
c) Una línea de programa puede tener un máximo de 255 caracteres, por tanto puede ocupar más de una línea en la pantalla; pulse RETUTN sólo al final de la línea de programa, antes de teclear el número de línea siguiente.
d) En la fase de diseño del programa utilice libremente las posibilidades que ofrece el editor de pantalla, descrito en el apartado anterior.
e) Para borrar una línea de programa es suficiente teclear su número seguido por RETURN.

3. Liste el programa tecleando LIST (o pulsando F4) y apretando RETURN y verifique la exactitud del listado.

4. Ejecute el programa tecleando RUN (o pulsando F5); puede ser (sobre todo al principio o en programas complejos) que aparezcan mensajes de error; los podrá interpretar con la ayuda del manual de su ordenador. Corríjalos y ejecute otra vez el programa. Repita este punto hasta obtener la ejecución sin errores y tal y como deseaba.

5. Salve el programa en el casete con la instrucción CSAVE. Escriba:

CSAVE nombre del programa

que, por supuesto, hay que terminar con RETURN; pulse las teclas RECORD y PLAY de la grabadora y espere a que el ordenador termine, lo que indicará escribiendo en la pantalla Ok. Tenga cuidado de verificar que también pulsó la tecla RECORD, porque sólo de esta forma el programa se grabará en la cinta. La operación puede requerir algunas decenas de segundos, en los que el ordenador parecerá inactivo, pero en realidad está trabajando como un loco. El volumen del casete debe estar alto; la posición exacta del control de volumen es cosa de experiencia; por lo tanto, no se desanime si las primeras veces el programa no se graba correctamente. Inténtelo otra vez con un volumen distinto.

6. Para verificar que el programa se ha grabado correctamente utilice el comando CLOAD:

CLOAD? nombre del programa

Antes de ejecutar esta instrucción deberá rebobinar la cinta. Tras pulsar RETURN accione el PLAY del casete. Si la grabación no ha efectuado correctamente, el ordenador contestará con el habitual Ok; en otro caso aparecerá el mensaje VERIFY ERROR (busque el error); compruebe entonces que la conexión con la grabadora es correcta, que sus pilas están en buenas condiciones de carga (o que el alimentador está enchufado), compruebe que el volumen de la grabadora no está en el mínimo o demasiado bajo para garantizar un buen nivel de grabación y, por último, recuerde si había pulsado las teclas de RECORD y PLAY simultáneamente. Después de haber verificado todo lo anterior y corregido todo lo erróneo, repita las operaciones oportunas.

7. El programa salvado en casete quedará memorizado de forma permanente: puede apagar el ordenador y volver después de una hora, un día o un año, seguro de encontrar el programa en el casete. Para transferirlo a la memoria, rebobine la cinta y teclee el comando CLOAD:

CLOAD nombre del programa

pulse RETURN y la tecla PLAY de la grabadora. El ordenador contestará con:

FOUND: nombre del programa

y empezará a cargar. Cuando termine de transferir a la memoria el programa, aparecerá el familiar Ok en la pantalla.

8. Ahora puede teclear RUN (o pulsar F5) para ejecutar el programa.

CAPITULO II

PRIMER CONTACTO CON EL BASIC MSX

En este capítulo vamos a darles información de carácter general sobre la forma de programar en BASIC MSX, válida también en algunos casos para otras versiones del BASIC.

La pantalla que emplee con su ordenador presentará, bajo control del BASIC MSX, un aspecto similar al mostrado en la figura 1.

Como puede ver, en la última línea aparecen cinco palabras que representan los cinco comandos asociados a las teclas de función F1-F5; pulsando la tecla SHIFT son sustituidos por los asociados a las teclas F6-F10 (color, cload, cont, list, run). En todo caso, el usuario puede modificar a su gusto tanto unos como otros. Con KEY OFF y KEY ON podemos inhibir o reactivar la visualización de los comandos en la última línea.

El mensaje Ok y la presencia del cursor parpadeante significan que el ordenador está en estado "comandos", es decir, a la espera de recibir una línea de programa BASIC, o un comando reconocido.

Las líneas de programa BASIC pueden tener un máximo de 255 caracteres, deben estar precedidas por un número de línea, y siempre se tienen que finalizar con RETURN para que el ordenador las acepte. En una misma línea de programa podemos incluir una o más instrucciones BASIC; en caso de ser varias deberán ir separadas por dos puntos (:).

Las instrucciones que escribamos sin número de línea delante se ejecutarán inmediatamente después del RETURN y el ordenador se olvidará después de ellas; es lo que se llama modo inmediato.



a)

```
MSX BASIC VERSION 1.0
Copyright 1983 by Microsoft
28815 Bytes free

OK

Color   auto   goto   list   run
```

b)

Fig. 1 · a) Equipo MSX en disposición para aceptar comandos. b) Detalle del mensaje en pantalla en la situación a).

Las instrucciones introducidas precedidas por un número de línea sólo se memorizan después del RETURN y van constituyendo lo que se llama programa residente en memoria. Tan sólo se ejecutan, según el orden creciente de su número de línea, cuando se da el comando RUN u otro cualquiera de ejecución; es lo que se conoce por modo diferido. Desde que recibe el comando RUN hasta que termina la ejecución, el ordenador se dice que está en estado de ejecución.

En memoria únicamente puede haber un programa BASIC a la vez, constituido por las instrucciones (numeradas) introducidas a través del teclado.

Si queremos borrar una instrucción es suficiente volver a teclear su número de línea seguido de RETURN; si deseamos sustituirla por otra distinta deberemos teclear su número de línea y, a continuación, la nueva instrucción; la sustitución será automática.

Las teclas de edición y algunas de control ofrecen las facilidades de un sencillo editor de pantalla, con el que es posible modificar, borrar e insertar caracteres en la pantalla; recordemos que después de cualquier modificación deberemos dar un RETURN para que el ordenador la grabe. Las funciones del editor de pantalla se describieron en el capítulo anterior.

Los errores que se pueden cometer al diseñar un programa pueden ser de dos tipos: formales y de contenido.

Los errores formales (o de sintaxis) se originan cuando se escribe una instrucción de forma equivocada, violando las reglas formales que definen el formato de dicha instrucción. En este caso el ordenador detecta el error y lo señala mediante un mensaje que aparece inmediatamente en la pantalla si estamos en modo inmediato, o durante la ejecución del programa si usamos el modo diferido; en este último caso la ejecución del programa se interrumpe en la instrucción en la que se ha encontrado el error, y el ordenador señala el número de línea.

Cada error tiene asociado un código, que corresponde a un número entero entre Ø y 59, para poder identificarlo. Haciendo adecuado uso de estos códigos el usuario experimentado puede interceptar el bloqueo que supone la aparición de un error y gestionarlo personalmente, con una rutina escrita por él mismo, impidiendo que el error interrumpa el programa. Para este tipo de aplicación se utilizan la instrucción ON ERROR GOTO y las variables ERR y ERL.

Los errores de contenido son los que, a pesar de no impedir la ejecución del programa, que resulta formalmente correcto, hacen que éste no realice adecuadamente la función o la aplicación para la que se ha escrito.

El BASIC MSX pone a disposición del usuario dos instrucciones (TRON y TROFF), que permiten visualizar los números de línea de las instrucciones según se van ejecutando en el programa, lo que resulta muy útil para verificar si la secuencia de ejecución del programa es la correcta, o si algún punto se desvía de lo que deseábamos.

DATOS NUMERICOS

— Constantes numéricas.

La representación adoptada para una constante numérica en el interior de un ordenador depende de un sufijo que determina su tipo:

[%] los números sin punto decimal, con o sin signo, comprendidos entre −32768 y +32767 seguidos del símbolo % se representan en dos bytes en complemento a 2, y se consideran enteros. Por ejemplo:

−5%
2ØØØØ%

[!] los números con o sin signo, con punto decimal o sin él, seguidos por el sufijo !, se representan en memoria con un máximo de 6 cifras significativas, en 4 bytes y se consideran reales en simple precisión.

6!
6.5!
899999.677777!

[#] los números con o sin punto decimal, con o sin signo, seguidos por el sufijo #, se representan en memoria con un máximo de 14 cifras significativas, en 8 bytes y se consideran reales en doble precisión; si tienen más de 14 cifras se aproximan redondeándolos a la decimocuarta cifra.

−5#
−5.8#
2ØØØØØØ#
9.12345678901234#

— Otras notaciones.

Una forma particular de introducir los números es la notación exponencial, en la cual el número se representa con una parte decimal y la letra (E o D) seguida por un número entero comprendido entre −64 y +62; el valor real del número se obtiene multiplicando la parte decimal por 1Ø elevado al exponente que sigue a la letra.

Los números en esta notación se consideran reales en simple precisión si la letra es E, en doble si la letra es D.

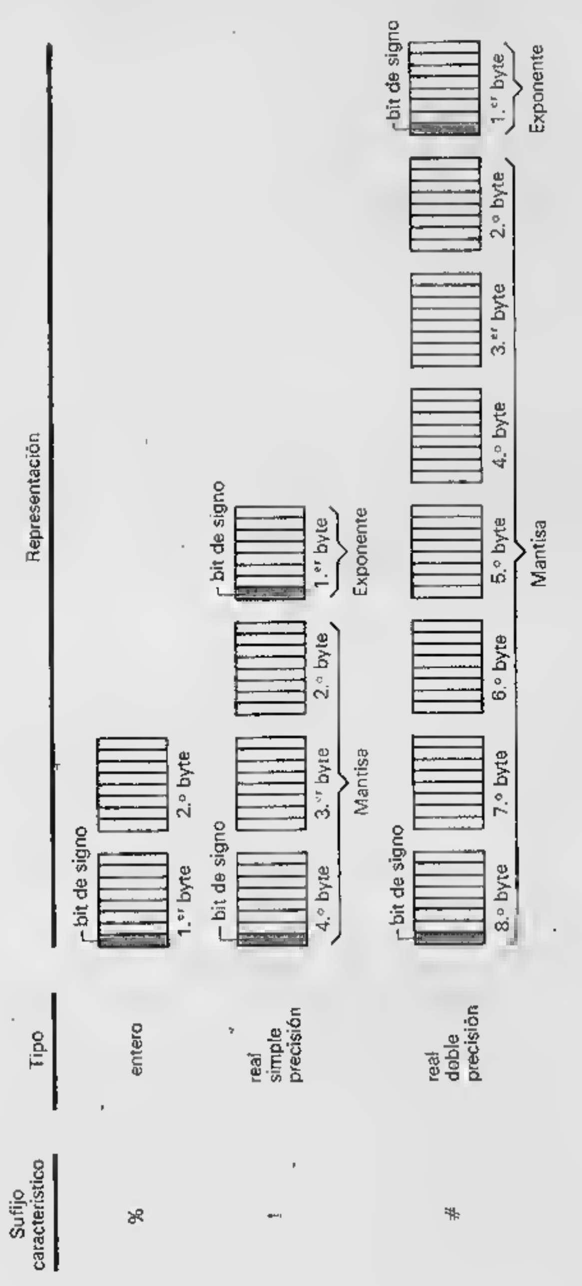


Fig. 2 - Forma en la que el ordenador representa internamente los tres tipos de constantes o variables numéricas.

0.5E2=50
-6.58E-2=-0.0658

Los números enteros se pueden representar además de en base decimal (10) en octal (8), en hexadecimal (16) y en binario (2), haciéndolos preceder, respectivamente, por

&O
&H
&B

Por ejemplo:

100
&0144
&H64
&B01100100 son expresiones equivalentes.

Los valores máximos representables son, respectivamente:

&O177777
&HFFFF
&B1111111111111111

— Variables numéricas.

En el BASIC MSX el nombre de una variable tiene que empezar con un carácter alfabético y puede ser tan largo como queramos, aunque sólo los dos primeros caracteres serán significativos para el ordenador. Esto significa que todos los nombres de variables que empiecen con dos caracteres iguales indicarán la misma variable a todos los efectos.

Esta particularidad puede llevar a errores de difícil identificación en la realización de un programa; por ejemplo, si, teniendo que definir los dos valores extremos para la variable X, los llamamos XMINIMO y XMAXIMO, el ordenador los considerará como una sola variable, con consecuencias inimaginables para el usuario. Es aconsejable, en estos casos, encontrar nombres distintos pero igualmente significativos para el usuario, como podrían ser XINFERIOR y XSUPERIOR.

Otra limitación es la que nos impide usar como nombres de variables las palabras reservadas del BASIC, es decir, los nombres-clave de las instrucciones y comandos.

Los criterios utilizados para determinar el tipo de las variables (enteras y reales de simple o doble precisión) son los mismos que hemos visto para las constantes, en base al sufijo y, además, para las variables es posible utilizar una definición de tipo explícita a través de las instrucciones:

DFINT
DEFSNG
DEFDBL

que definen, respectivamente, las variables como enteras, de simple y de doble precisión.

En caso de doble y contradictoria definición de una variable, mediante sufijo e instrucción declarativa, el tipo determinado por el sufijo prevalecerá respecto a la instrucción de definición explícita. Algunas funciones de sistema nos permiten efectuar conversiones de variables de un tipo a otro.

DATOS ALFANUMERICOS

— Constantes de cadena.

Son datos alfanuméricos, es decir, compuestos por cifras, caracteres alfabéticos y otros símbolos, que contienen un máximo de hasta 255 caracteres reconocibles por el ordenador y que deben ir delimitados por comillas. Por ejemplo:

"AMIGO"
"AMOR"
"ORDENADOR"

Se llama cadena nula a la cadena

""

constituida por dos parejas de comillas sucesivas.

— Variables de cadena.

Son variables destinadas a contener valores alfanuméricos. Se identifican con un nombre que obedece a las mismas reglas enunciadas para las de las variables numéricas y que, además, tiene que tener como sufijo el símbolo $

A$
PABLO$
TITO$
TOTAL$

El espacio disponible en la memoria del ordenador para este tipo de variables es de 200 caracteres, pero es posible cambiar este valor con la instrucción CLEAR.

Los datos alfanuméricos, sean variables o constantes, no se pueden usar en cálculos aritméticos, pero hay instrucciones específicas que nos permiten manejarlos con comodidad.

ARRAY

El tipo de elementos que puede contener un array sigue las mismas reglas que hemos visto para las variables normales. Por tanto:

A%(I)	es el i-ésimo elemento de un array de números enteros;
A!(I)	es el i-ésimo elemento de un array de números reales en simple precisión;
A(I) o A#(I)	es el i-ésimo elemento de un array de números reales en doble precisión;
A$(I)	es el i-ésimo elemento de un array de cadenas de caracteres.

EXPRESIONES ARITMETICAS

Están constituidas por constantes, variables, funciones aritméticas, paréntesis redondos y operadores aritméticos. Los operadores aritméticos están detallados a continuación en orden de prioridad decreciente:

^	exponenciación
*	multiplicación, división
\	división entera
mod	resto de una división entera
+-	adición, sustracción.

El ordenador lleva a cabo las expresiones ejecutando primero las operaciones con prioridad más alta y luego las demás; las de una misma prioridad las atiende según las vaya encontrando al recorrer la expresión de izquierda a derecha. Los paréntesis alteran los niveles de prioridad convencionales reseñados: las operaciones entre paréntesis se ejecutan con prioridad absoluta respecto a las demás, partiendo, naturalmente, de las más internas. Así 3*5+8 da como resultado 23, pero si ponemos 3*(5+8), el valor obtenido será 39.

Puede haber varios niveles de paréntesis anidados:

((3+8)*5-3)/4 (resultado=13)

Según cual sea el tipo de los operadores (constantes o variables) que aparecen en una expresión, así será el resultado. En la figura 3 se indican las posibles combinaciones.

OPERANDOS		RESULTADO
OP%	OP%	R%
OP%	OP!	R!
OP%	OP#	R#
OP%	OP	R
OP%	OP$	ERROR
OP!	OP!	R!
OP!	OP#	R#
OP!	OP	R
OP!	OP$	ERROR
OP#	OP#	R#
OP#	OP$	ERROR
OP	OP	R

Fig. 3 - Tipo del resultado de una expresión en base al tipo de sus operandos.

EXPRESIONES DE CADENA

El operador + se puede utilizar también para añadir una cadena a otra. Por ejemplo, con:

```
10 PRINT"calle "+"mayor"
```

obtendríamos la expresión: calle mayor.

EXPRESIONES RELACIONALES

Están constituidas por dos operandos, numéricos o de cadena, que se comparan entre sí por medio de los llamados operadores relacionales, que son, concretamente:

=	igual
<	menor
>	mayor
><, <>	distinto
<=, =<	menor o igual
>=, =>	mayor o igual

Los dos operandos usados tienen que ser homogéneos entre sí, o sea, dos números o dos cadenas. Por lo que se refiere a las últimas, se dice que una cadena es menor que otra cuando la precede en orden alfabético, y que es mayor cuando la sigue. Así DAMA es menor que TORRE y mayor que ALFIL.

Recordemos a este respecto que en las cadenas los espacios cuentan como cualquier otro carácter.

El resultado de la comparación determina el valor de la expresión relacional:

1 si la condición es verdadera
Ø si la condición es falsa

EXPRESIONES LOGICAS

Están constituidas por constantes, variables, funciones aritméticas o de cadena y por los operadores lógicos. Estos actúan sobre los operandos (transformados en números enteros de 16 bits) bit por bit.

La figura 4 representa a los operadores lógicos y su "tabla de la verdad", que indica el valor que depositan en el bit i-ésimo del resultado al tratar los bits correspondientes de los operandos.

MAPA DE MEMORIA

La figura 5 reproduce el mapa de memoria del ordenador Sony Hit Bit.

Como puede ver, la zona comprendida entre las direcciones HØØØØ y H7FFF (32767) está ocupada por el intérprete BASIC y es de sólo lectura (hay una ROM). La memoria situada entre las direcciones HF38Ø (62336) y HFFFF (65535) es utilizada por el sistema para sus variables internas, por lo que debe ser de tipo RAM.

La RAM destinada al usuario está entre las direcciones HØØØØ (49152) y HF37F (62335) o entre las direcciones H8ØØØ (32768) y HF37F (62336), dependiendo de las dimensiones de la RAM instalada (16K y 32K, respectivamente).

En la figura 5.b se detalla el mapa de la memoria de usuario. En el área de programa reside el programa actual, o, mejor dicho, las instrucciones, con su número de línea, del programa actual; en el área de variables residen las variables de tipo numérico y los punteros a las zonas donde están memorizadas las variables de cadena; en el área de variables con índice residen los arrays numéricos y los punteros a las zonas donde están memorizados los arrays de cadena.

X	NOT X
1	0
0	1

– NOT

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

∧ AND

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

∨ OR

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

⊻ XOR

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

= EQV

Fig. 4 - Operadores lógicos y su tabla de verdad respectiva.

El stack es una zona de memoria gestionada por el sistema operativo para memorizar informaciones temporalmente, en particular las direcciones de reentrada desde las subrutinas. El área de las variables de cadena contiene las variables de cadena, sencillas y con índice. Para acabar, el área de gestión de los ficheros es una zona de memoria utilizada por el sistema para operaciones de entrada y salida con los ficheros.

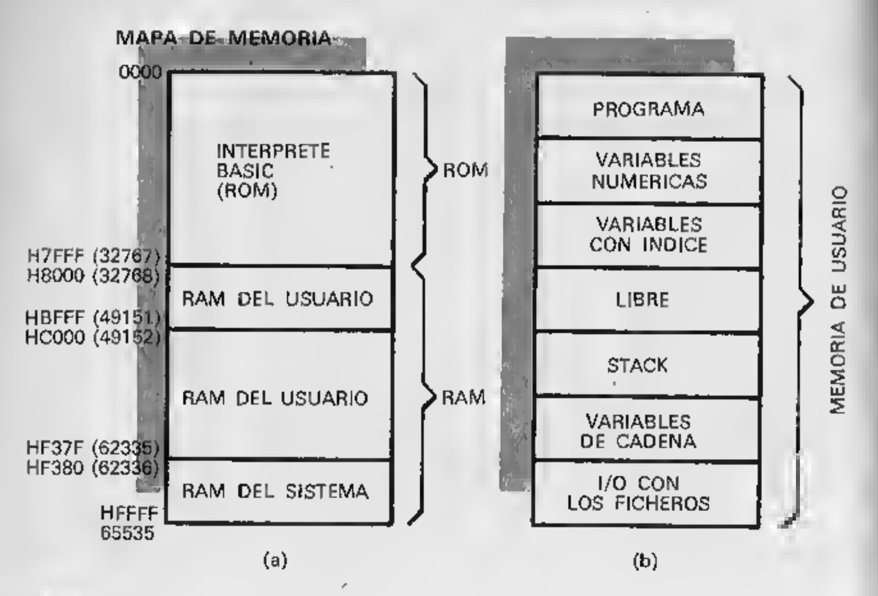


Fig. 5 - Mapa de memoria.

En los capítulos siguientes se describen la mayor parte de las instrucciones del BASIC MSX, divididas por grupos funcionales. Cada instrucción está constituida por una o más palabras clave y uno o más operandos; en la presentación de las instrucciones hemos intentado indicar dichos operandos con nombres que aclaren el significado o la función. Así, cuando un operando está indicado con el término "número" o "constante" significa que tiene que ser una constante, numérica o de cadena según lo especificado, y no puede ser una variable o una expresión; en todos los demás casos se entiende que el operando puede ser cualquier expresión del tipo adecuado.

Los operandos enmarcados por paréntesis cuadrados [ ] son opcionales, lo que significa que se pueden omitir, en cuyo caso el sistema los sustituye por valores normalizados conocidos como valores de defecto (default).

Cuando una instrucción tiene más de un operando, éstos se separan con una coma; si se omite alguno hay que señalar su ausencia con dos comas seguidas, a menos que se trate de operandos terminales.

NOTA: Los paréntesis redondos, cuando los haya, forman parte de la sintaxis de la instrucción y, consecuentemente, hay que teclearlos.

CAPITULO III

PRINCIPALES INSTRUCCIONES DEL BASIC MSX

Instrucciones y comandos de escritura y puesta a punto del programa

AUTO [número de línea inicial] [, incremento]

Genera automáticamente los números de línea del programa que se quiere introducir desde teclado, a partir de un valor inicial dado y con el incremento especificado.

Los valores de defecto son Ø para el valor inicial y 1Ø para el incremento; si ambos se omiten, la primera línea será numerada con un 1Ø. Cuando en la memoria ya está presente una línea con el número generado, éste aparece con un asterisco; si no se quiere modificar la instrucción existente pulse RETURN; de otra forma teclee la nueva instrucción, ésta sustituirá automáticamente a la anterior.

Para salir de esta situación de generación automática bastará pulsar CTRL+C o STOP.

DELETE [número de línea inicial] [- número de línea final]

Borra todas las instrucciones comprendidas entre los números de línea especificados, inclusives.

Hay que dar por lo menos uno de los dos números; dependiendo de los valores que escribamos será posible borrar todas o parte de las líneas del programa.

DELETE

Borra la última línea visualizada o ejecutada.

LIST [número de línea inicial] [- número de línea final]

Visualiza en la pantalla todas las líneas de programa comprendidas entre los dos números especificados.
Dependiendo de los valores dados a los dos números es posible visualizar todas o parte de las líneas del programa.

LIST

Visualiza la última línea visualizada o ejecutada.

LLIST [número de línea inicial] [- número de línea final]

Realiza la misma función que LIST, pero el listado lo saca por la impresora. Si ésta no está conectada el ordenador se bloqueará y no aceptará ningún otro comando desde el teclado; para desbloquearlo habrá que pulsar CTRL+STOP o bien conectar la impresora.

NEW

Borra todas las líneas de BASIC presentes en la memoria y pone el ordenador en estado comandos.
Los programas en código máquina que pudieran estar presentes en la memoria no se borran.

RENUM [nuevo número inicial] [,antiguo número inicial] [, incremento]

Realiza la renumeración automática de todas las líneas del programa que se encuentra en memoria a partir de la línea que tiene el señalado como "antiguo número inicial"; éste se sustituye por el "nuevo número inicial" y las demás líneas, hasta el final, se numeran siguiendo el incremento especificado. Todas las referencias a números de líneas que aparecen en las instrucciones se modifican consecuentemente.
El valor de defecto para el "nuevo número inicial" y para el incremento es 10, para el "antiguo número inicial" es el número de la primera línea del programa en memoria.

REM [comentario]

Identifica las instrucciones de comentario, que el ordenador no ejecuta y que sirven para documentar y hacer más legible el programa.

KEY LIST

Visualiza las cadenas de caracteres asociadas a las teclas de función F1-F10.

Instrucciones declarativas

MAXFILES=expresión

Limita el número máximo de ficheros que se pueden abrir durante la ejecución del programa al valor dado por "expresión", entre 0 y 15. Si es 0 sólo puede realizarse SAVE y LOAD.

CLEAR [dimensión área variables cadena] [, dirección más alta]

Pone a cero las variables numéricas, anula las variables de cadena, define las dimensiones del área de memoria asignada a las variables de cadena (en bytes) y especifica la dirección más alta de memoria utilizable en BASIC.
Por defecto se utilizan los valores del mapa de memoria estándar o los definidos con el último CLEAR. El primer valor no se puede omitir si está presente el segundo.

DIM nombre variable (máx. valor índice 1 [, máx. valor índice 2,...])

Cuando los índices de un array o matriz puedan tener un valor superior a 10, hay que declararlo con la instrucción DIM, con la que determinamos el nombre de la variable (que definirá también su tipo) y los valores máximos de cada índice.
Los arrays pueden tener un número cualquiera de índices, siempre supeditados a la capacidad de la memoria.
Con una sola instrucción DIM se pueden declarar tantos arrays como quepan en la línea.
El límite máximo para los índices de matrices no dimensionadas se fija automáticamente en 10.
Recordemos que los índices varían desde 0 hasta el valor máximo, lo cual quiere decir que una matriz no dimensionada, o di-

mensionada con la instrucción DIM A(1Ø), por ejemplo, contiene 11 elementos: A(Ø), A(1),..., A(1Ø).

Un array no puede aparecer en el mismo programa en más de una instrucción DIM, a menos que las anteriores se hayan borrado con la instrucción ERASE (verla en este mismo apartado). Un array y una variable simple pueden tener el mismo nombre sin crear ambigüedad, y se tratan como variables distintas (como vimos en el apartado Mapa de Memoria del capítulo anterior sus zonas de memoria y sus punteros, según el caso, son distintas).

DEFINT car. alfabético 1 [– car. alfabético 2,...]

Define como variables enteras todas las que empiezan con una letra comprendida entre dos caracteres alfabéticos especificados o con una letra igual a un carácter determinado.

Recordemos que los sufijos de declaración del tipo %, !, #, tienen una prioridad superior a la de las instrucciones de declaración, como DEFINT.

DEFSNG car. alfabético 1 [– car. alfabético 2,...]

Como DEFINT, pero para las variables reales en simple precisión.

DEFDBL car. alfabético 1 [– car. alfabético 2,...]

Como DEFINT, pero para las variables reales en doble precisión.

DEFSTR car. alfabético 1 [– car. alfabético 2,...]

Como DEFINT, pero para las variables de cadena.

DEF FNnombre función [(parámetro 1, parámetro 2,...)]= expresión

Define una función, aritmética o de cadena, que se puede utilizar cuando se quiera a lo largo del programa simplemente llamándola por nombre.

Por ejemplo:

```
10 DEF FNFUNCION(x,y)=(x*y)+(x/y)
20 A=FUNCION(8,10)-25
```

El nombre de la función puede ser cualquiera de los válidos para las variables, con las mismas reglas para decidir el sufijo del tipo.

Los parámetros son los nombres de las variables simbólicas que se emplean para definir la función. Cuando la función es llamada en el transcurso del programa, estas variables simbólicas (también conocidas como "mudas") se sustituyen por las variables o valores efectivos con los cuales se quiere calcular la función.

Por ejemplo:

```
10 DEF FNA(I,J)=I+J
20 B=5:C=8
30 D=FNA(B,C)
```

ERASE nombre array 1 [, nombre array 2,...]

Borra del área reservada a las variables con índice las matrices especificadas, anulando las posibles instrucciones DIM anteriores, después de lo cual se pueden volver a utilizar los mismos nombres de variables para matrices y pueden aparecer, por tanto, en otra instrucción DIM.

KEY número de la tecla de función, "cadena de caracteres"

Asocia a la tecla de función especificada una cadena de como máximo 14 caracteres. Los caracteres pueden ser alfanuméricos, de control o de cualquier otro tipo. Cuando se pulse la tecla de función, el ordenador pondrá en pantalla los caracteres asociados, como si los hubiéramos introducido nosotros por teclado.

Por ejemplo:

```
KEY 1, "NEW"
```

Cuando se apaga el ordenador o se pulsa la tecla RESET se pierden estas cadenas y las teclas de función asumen otra vez sus valores estándar o de defecto.

Instrucciones de Entrada/Salida y asignación

DATA constante [, constante,...]

Conserva los valores, numéricos o alfanuméricos, que asignaremos a las variables enumeradas en las sucesivas instrucciones READ (verla a continuación), separadas entre sí por comas. El tipo de cada constante tiene que ser compatible con el de la variable a la que se tiene que asignar.

Las constantes de cadena que contengan comillas, dos puntos o espacios iniciales y finales tienen que estar entre comillas.

Por ejemplo:

100 DATA PEPEJUAN,"1 DE ENERO", "18:35:40",25,43

READ variable 1 [, variable 2,...]

Lee los datos especificados en las instrucciones DATA y los asocia a las variables de su lista en el mismo orden en que aparecen: la primera instrucción READ del programa empieza a leer los datos de la primera DATA y cuando se agotan (en esta o en READ sucesivas) los datos contenidos en ella, siguen con el primer dato de la siguiente DATA, y así hasta el final.

El tipo del dato tiene que ser compatible con el de la variable a la que se tiene que asignar.

RESTORE [número de línea]

Modifica el orden de lectura de las DATA que leen las READ. La primera instrucción READ que ejecute el programa después de una instrucción RESTORE empieza a leer los datos de la primera DATA que se encuentre a partir del número de línea especificado en la RESTORE o, si éste faltara, de la primera DATA del programa. A partir de ese momento el proceso es el mismo que vimos para READ y DATA, leyéndose sucesivamente los datos de las DATA siguientes a la especificada.

INPUT ["mensaje";] variable 1 [, variable 2,...]

Cuando el ordenador encuentra esta instrucción, visualiza el mensaje, si existe, y un signo de interrogación (?) e interrumpe la ejecución del programa en espera de que se introduzcan desde el terminal los valores que hay que asignar a las variables especificadas en la lista.

Es posible introducir los datos cada uno en una línea, si después de pulsarlo damos un RETURN, o uno tras otro si los separamos por comas y con un RETURN final.

En todo caso, si el número de datos proporcionados es inferior al de las variables de la lista, el ordenador pondrá ?? y seguirá esperando que el usuario introduzca los que restan. Si, en cambio, escribimos más de los necesarios, aparecerá el mensaje

?Extra ignored

comunicándonos que los datos de más son ignorados, y la ejecución del programa continuará. El tipo de cada dato tiene que ser

compatible con el de la variable correspondiente; si no fuera así, el ordenador escribiría:

?Redo from start

LINE INPUT ("mensaje";) variable de cadena

Cuando el ordenador encuentra esta instrucción visualiza el mensaje, si lo hay, y un signo de interrogación y se para; todos los caracteres que el usuario introduzca a partir de este momento hasta el RETURN, incluyendo eventuales signos de puntuación, se asignan a la variable de cadena especificada.

PRINT expresión 1 [separador 1 expresión 2 separador 2...]

Visualiza en la pantalla los valores de las expresiones especificadas, que pueden ser variables, constantes u operaciones. Las constantes de cadena deberán aparecer entre comillas.

Si una expresión va seguida por una coma, la siguiente aparecerá al principio de la próxima zona de pantalla (recordemos que la pantalla está dividida en zonas de 14 caracteres); si una expresión está seguida por un punto y coma, la siguiente aparecerá a continuación, separada tan sólo por un espacio.

Lo dicho sirve también para el último y el primer elemento de dos PRINT sucesivos: si la lista de un PRINT termina sin ningún separador, se ejecuta automáticamente un CRLF (retorno de carro y línea arriba) y el PRINT siguiente empieza a escribir sus resultados desde el principio de la línea siguiente.

Los números negativos se visualizan con el signo, mientras que en los positivos el signo se omite y se sustituye por un espacio.

? expresión 1 [separador 1 expresión 2...]

Equivale a todos los efectos al PRINT visto antes.

PRINT USING "formato"; expresión 1 [expresión 2...]

Visualiza los valores de las expresiones según el formato especificado.

El formato está indicado con símbolos especiales. A continuación se detallan los distintos formatos y su significado.

imprime únicamente el primer carácter de cada cadena;

& imprime un número de caracteres de la cadena igual al número de espacios comprendidos entre dos ampersands (&) más dos; si supera la longitud de la cadena añade espacios.

@ imprime la cadena tal cual.

— FORMATOS NUMERICOS

== imprime tantas cifras como símbolos # pongamos a la izquierda y a la derecha de la coma. Las cifras de la parte entera se justifican a la derecha, las de la parte decimal se justifican a la izquierda, seguidas por Ø si son menos que los # especificados; si son más, el número se redondea. El signo + se omite, mientras que el – está considerado como una cifra más, lo cual se tiene que tomar en cuenta a la hora de decidir el número de sostenidos (#) de la parte entera.

\+ imprime el signo del número, sea positivo o negativo, delante o detrás según donde situemos el +.

– sólo puede ir en el formato después de la cadena de #, hace que los números negativos se impriman seguidos por un –, y los positivos, por un espacio en blanco.

** añade los asteriscos que se necesiten delante del número para llenar el campo, cuya dimensión total viene dada por el número de # y *.

$$ imprime un $ antes del dato numérico;

, tiene que ser especificada antes del punto decimal; provoca que la parte entera del número se imprima con una coma separando cada tres cifras (esto es así por emplear los americanos una notación donde el "." y la ",", tienen la interpretación contraria a la que se les suele dar en Europa);

↑↑↑↑ permite la notación exponencial; reserva espacio para los 4 caracteres del exponente.

Si en cualquiera de éstos se rebasa el campo, el BASIC intentará aproximarse lo más posible al formato dado, pero imprimirá un % delante para indicar esta circunstancia.

LPRINT expresión 1 [separador 1 expresión 2 separador 2...]

Igual que PRINT, pero para producir la salida a impresora.

LPRINT USING "formato"; expresión 1 [expresión 2...]

Como PRINT USING, pero para producir la salida a impresora.

[LET] variable=expresión

Es la típica instrucción de asignación del BASIC; la palabra clave LET se puede omitir, y es lo que se hace casi siempre.

El valor obtenido calculando la expresión situada a la derecha del signo = se asigna a la variable que se encuentra a la izquierda. El tipo de la expresión tiene que ser compatible con el de la variable; en el caso de magnitudes numéricas, el resultado de la expresión se convierte al tipo de la variable.

MID $(X$,M [,N])=Y$

Sustituye a partir del carácter M, inclusive, de X$ tantos caracteres como indique N por los correspondientes N primeros de Y$ o por toda la cadena Y$ si no se especifica N.

M y N pueden ser expresiones numéricas de cualquier complejidad, cuyo valor esté entre Ø y 255.

Por ejemplo:

```
10 A$="Sonrojo":B$="risa"
20 MID$ (A$, 4,4)=B$
```

produce un nuevo A$="Sonrisa"

SWAP variable, variable

Intercambia los valores de dos variables. Recordemos que no se pueden intercambiar directamente los valores de dos variables con las instrucciones

```
A=B
B=A
```

porque la primera hace que perdamos el valor original de A y la segunda, entonces, deja a B sin modificar.

Si quisiéramos hacerlo de esta forma habría que usar lo que se llama una variable auxiliar:

```
C=A
A=B
B=C
```

Instrucciones de control

CLS

Borra todo el contenido de la pantalla.

RUN [número de línea]

Ejecuta el programa presente en la memoria a partir del número de línea especificado; por defecto, si no lo especificamos, el ordenador toma el número de la primera línea de programa.

Si en el transcurso de la ejecución se encuentra cualquier error la ejecución termina y el ordenador genera un mensaje de error con la indicación del número de línea donde se produjo; de lo contrario, la ejecución acaba cuando el programa lo determina. En cualquiera de los casos el sistema vuelve al estado "comandos", apareciendo el mensaje Ok y el cursor intermitente.

La ejecución se puede interrumpir en cualquier momento con la tecla STOP o con CTRL+STOP; en el segundo caso hay que dar el comando CONT para seguir, mientras que en el primero se puede pulsar simplemente STOP.

STOP

Usada como instrucción en el programa hace que cuando el ordenador la encuentra, interrumpe la ejecución visualizando el mensaje Break in xxx, donde xxx sustituye el número de línea de la instrucción STOP. Para seguir habrá que usar el comando CONT. A diferencia con END, la instrucción STOP no se ocupa de cerrar los ficheros.

CONT

Vuelve a activar la ejecución de un programa interrumpida por la tecla STOP (o la instrucción) o CTRL+STOP, a partir del número de línea siguiente al de la interrupción; hace una excepción el caso de que la interrupción se produjera en una instrucción INPUT, en cuyo caso la ejecución comienza a partir de esa misma instrucción.

END

Se tiene que usar al final del programa principal y antes de que comiencen las subrutinas (si las hemos escrito detrás) para evitar que se vuelvan a ejecutar a continuación del programa principal y provoquen errores. Cuando se encuentra se detiene la ejecución y se cierran todos los ficheros que estuvieran abiertos, apareciendo en pantalla el Ok y el cursor intermitente.

También se puede poner en cualquier otro punto del programa para señalar el final físico de una ramificación.

Se puede usar en lugar de STOP, aunque no ocurre lo mismo al revés. La ejecución puede volver a empezar con una instrucción RUN o GOTO, pero no con la instrucción CONT.

TRON

Siempre que el ordenador no esté en modo gráfico, este comando visualiza los números de línea de las instrucciones que se van ejecutando. Por lo general se usa para la depuración del programa, es decir, para la búsqueda de los errores que impiden que un programa se desarrolle correctamente, lo que en inglés se denomina "debugging". El BASIC MSX permite realizar esta operación con el proceso TRACE, que se activa mediante TRON (TRACE ON). Lo normal es usarla en modo inmediato.

TROFF

Anula la instrucción TRON, desactivando el proceso TRACE (TRACE OFF). Se suele usar en modo inmediato.

FOR variable=valor inicial TO valor final [STEP incremento]

Las instrucciones que se encuentran entre esta instrucción y el siguiente NEXT se ejecutan un número de veces determinado por los valores especificados, en lo que se llama bucle o ciclo.

Cuando el programa encuentra esta instrucción memoriza los valores actuales del valor inicial y el final (que pueden ser variables o expresiones) y asocia el primero a la variable índice. Al hallar el NEXT retorna al FOR sumando el valor de "incremento" positivo o negativo a la variable índice. La ejecución termina cuando la variable índice supera el valor final (el bucle no se realiza entonces).

El valor de defecto para el incremento es 1.

Varios ciclos pueden estar anidados uno dentro de otro, pero no pueden, por razones lógicas, intersecarse. Así:

Correcto	Incorrecto
FOR I=1 TO 20	FOR L=0 TO 5
FOR J=8 TO 0 STEP -1	FOR P=-1 TO 2
NEXT J	NEXT L
NEXT I	NEXT P

NEXT variable

Cierra el bucle que comenzó en su FOR asociado.

GOSUB número de línea

Es la instrucción de llamada a subrutina. Cuando el ordenador la encuentra transfiere el control a la instrucción con el núm

ro de línea especificado (entrada a la subrutina) y la ejecución sigue a partir de este punto hasta encontrar un RETURN; entonces el control retorna a la instrucción con el número de línea inmediatamente siguiente al de la llamada, a menos que en el RETURN se especifique otro distinto.

Una subrutina puede a su vez llamar a otra, pero hay que tener mucho cuidado a la hora de anidar subrutinas para evitar errores de programa.

Generalmente, las subrutinas se escriben detrás del programa principal, separándolas de éste por un END.

RETURN [número de línea]

Transfiere el control a la instrucción que se encuentra en el número de línea especificado; si falta el número de línea el control se devuelve a la instrucción inmediatamente siguiente a aquella desde la cual se llamó a la subrutina en la que se encuentra el RETURN.

GOTO número de línea

Es la instrucción de salto incondicional; transfiere el control a la instrucción que se encuentra en el número de línea especificado.

Se puede usar en modo inmediato, en lugar de RUN, para ejecutar un programa a partir de una línea intermedia.

IF expresión THEN instrucción [ELSE instrucción]

Si la expresión es verdadera, es decir, si asume un valor distinto de Ø, se ejecuta la instrucción que sigue a THEN y luego se pasa a la instrucción que se encuentra en la línea siguiente. Si es falsa se ejecuta la instrucción que sigue a ELSE y luego se pasa a la línea siguiente. En caso de que faltara el ELSE y la expresión fuera falsa se pasaría directamente a la línea siguiente.

Si la instrucción que sigue al THEN es un GOTO, una de las palabras clave (THEN o GOTO) se puede omitir, especificando solamente el número de línea. Así, son equivalentes:

```
IF (A=B) THEN 21Ø
IF (A=B) GOTO 21Ø
```

Tanto después del THEN como después del ELSE puede haber varias instrucciones en la misma línea, separadas por dos puntos; en este caso se ejecutan o se saltan todas en bloque antes de pasar a la línea siguiente.

Después del THEN y del ELSE puede haber otras instrucciones IF, realizándose así ramificaciones múltiples.

ON expresión GOTO número de línea 1 [, número de línea 2,...]

Si "n" es el valor de la expresión aritmética que sigue a ON, el control se transfiere al número de línea que ocupa la posición n-é-sima después de GOTO.

Si "n" vale Ø o un valor superior a la cantidad de números de línea especificados, el control pasa a la instrucción siguiente. Si "n" es menor que Ø o mayor que 255 se genera un mensaje de error. Si "n" no es entero, se trunca al valor entero sin redondeo.

ON expresión GOSUB número de línea 1 [, número de línea 2,...]

Igual que ON GOTO, con la diferencia de que el control pasa a una subrutina.

CAPITULO IV

GRAFICOS



a unidad elemental de visualización en la pantalla se llama pixel (contracción de picture element), y constituye la imagen más pequeña visible, por lo que viena a equivaler a un punto, término con el que también se le puede llamar.

La pantalla estándar MSX está constituida por 256 puntos horizontales y 192 verticales. Podemos imaginarla como si estuviera formada por una serie de planos, tal y como muestra la figura 1.

Como puede ver, todos los planos van superpuestos y su unión es precisamente lo que veremos en la pantalla. Delante del plano de fondo encontramos el primer plano, en el cual es posible visualizar textos compuestos por caracteres alfanuméricos, o dibujar líneas y gráficos. Los conocidos como bordes son dos zonas marginales, una en la parte superior y otra en la parte inferior de la pantalla. Estas tres superficies: primer plano, fondo y bordes se pueden colorear de forma distinta con la instrucción COLOR que veremos después; para cada uno es posible elegir entre 16 colores, identificados por un código de Ø a 15.

Existen además otros 32 planos, numerados del Ø al 31, en los que podemos hacer aparecer una imagen definida por el usuario (sprite) y moverla a nuestro gusto.

Existen cuatro modos operativos distintos en los que el MSX permite que trabajemos con la pantalla, seleccionables mediante la instrucción SCREEN e identificables con un número que varía entre Ø y 3:

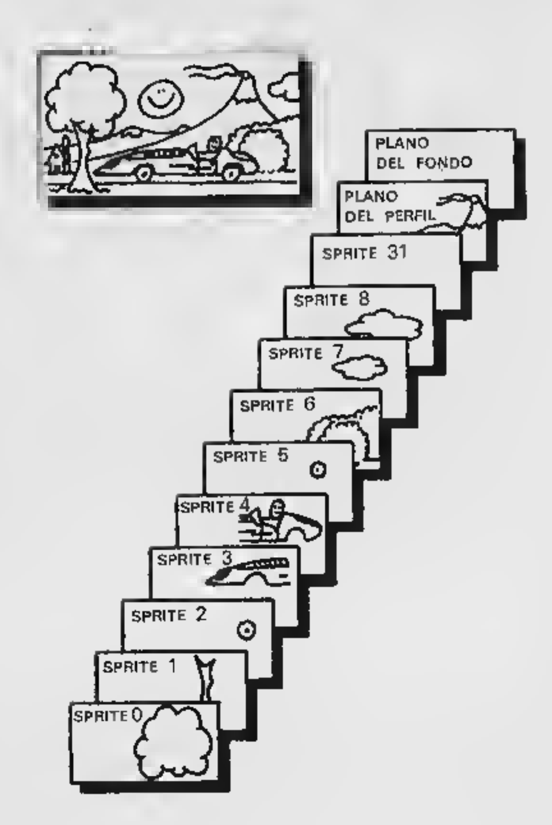


Fig. 1 · Planos de la pantalla.

SCREEN Ø
SCREEN 1
SCREEN 2
SCREEN 3

En los modos Ø y 1 (pantallas de texto) sólo se pueden visualizar caracteres alfanuméricos pertenecientes al set de caracteres del BASIC MSX. En el Apéndice C están detallados todos estos caracteres con su correspondiente código ASCII.

En modo Ø cada carácter tiene una anchura de 6 puntos horizontales, suficiente para casi todos los caracteres excepto para algunos, típicos del MSX, que tienen una anchura de 8 puntos, con lo cua no se visualizarán correctamente en el modo o pantalla Ø.

En el modo 1, en cambio, los caracteres se visualizan con 8 puntos horizontales, lo que hace que todos los caracteres se visualicen correctamente y la legibilidad sea mejor. Naturalmente en este modo se visualiza un menor número de caracteres por línea.

En ambos modos la cantidad de caracteres visualizados por línea se establece con el comando WIDTH, que veremos en seguida.

En los modos 2 y 3 (pantallas gráficas) se pueden realizar dibujos y gráficos con las numerosas instrucciones que describiremos a lo largo de este capítulo. En el modo 2 los gráficos se trazan por puntos (alta resolución) y en el 3 por cuadraditos de 4×4 puntos (baja resolución). En estos modos no es posible, en principio, visualizar caracteres con las instrucciones normales; aunque es posible lograrlo con un simple truco que consiste en considerar la pantalla como un dispositivo capaz de recibir y grabar ficheros; esta técnica se explica en el capítulo 6, dedicado a los ficheros.

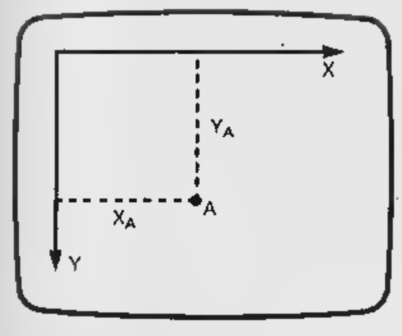
En el modo 2 la resolución es más alta pero, como contrapartida, tenemos una menor flexibilidad por lo que se refiere al color. Efectivamente, cada línea de pantalla se considera, a efectos del color, dividida en grupos de 8 pixels adyacentes y en el interior de cada grupo sólo puede haber un color; si asignamos colores distintos a puntos del mismo grupo con instrucciones gráficas, el grupo se colorea con el color asignado en último lugar.

En el modo 3, en cambio, la resolución es menor desde el punto de vista gráfico, ya que, como hemos dicho, la unidad elemental de trazado de gráficos es un bloque de 4×4 puntos, pero, por la misma razón, en un grupo de 8 puntos horizontales puede haber dos colores distintos. Por esta razón el modo 3 se llama también multicolor.

En estos dos modos las coordenadas de un punto se dan en número de pixels horizontales (de Ø a 255) y verticales (desde Ø a 191). Naturalmente, en modo 3, todos los valores que están comprendidos en un mismo cuadradito de 4×4 determinan igual bloque y son, por tanto, equivalentes. Los números que determinan las coordenadas se refieren a un sistema de ejes x-y cuyo origen está en el punto superior izquierdo de la pantalla, el eje "x" es horizontal, orientado hacia la derecha, y el eje "y" es vertical, orientado hacia abajo (Fig. 2a).

Las instrucciones gráficas, con la opción STEP, permiten dar las coordenadas relativas, es decir, no referidas al origen del sistema de referencia (coordenadas absolutas), sino al punto alcanzado con la última instrucción gráfica (coordenadas relativas); este punto se indica como origen relativo (Fig. 2b).

En todos los modos, excepto en el Ø, es posible visualizar uno o más sprites sobre 32 planos distintos, realizando así superposiciones de imágenes.



(a) Coordenadas absolutas del punto A

(b) Coordenadas absolutas del punto A y relativas del B y el C



Fig. 2 - Coordenadas absolutas y relativas.

El *sprite* es una imagen definida por el usuario en una zona de 8×8 o 16×16 pixels, que puede ser de tamaño normal o doble (ampliada), como muestra la figura 3.

El tamaño del sprite, en número de puntos, y la dimensión de la zona se definen con la instrucción SCREEN.

La imagen se realiza individualizando en el interior de la zona de 8×8 o 16×16 los puntos que la componen y asignándoles un color determinado. La mecánica de construcción de un sprite es un poco larga, pero sencilla. Para un sprite de 8×8, como el de la figura 4, sería:

1. A cada línea del sprite se le hace corresponder un byte (8 bits), cuyo valor binario lo obtenemos al poner un 1 en los bits que corresponden a los puntos activados del sprite y Ø en los demás.
2. Se convierte el número binario que se obtiene a decimal o hexadecimal.
3. Se utilizan los valores obtenidos como argumentos de funciones CHR$ que se suman ordenadamente comenzando por la primera línea superior.
4. La expresión obtenida se asigna a la variable SPRITE$(I), donde I es el número que a partir de este momento identifica la forma del sprite así definida.

Los sprites de 16×16 puntos se definen de forma idéntica, subdividiéndolos en cuatro grupos de 8×8 y describiéndolos en el orden que muestra la figura 5.

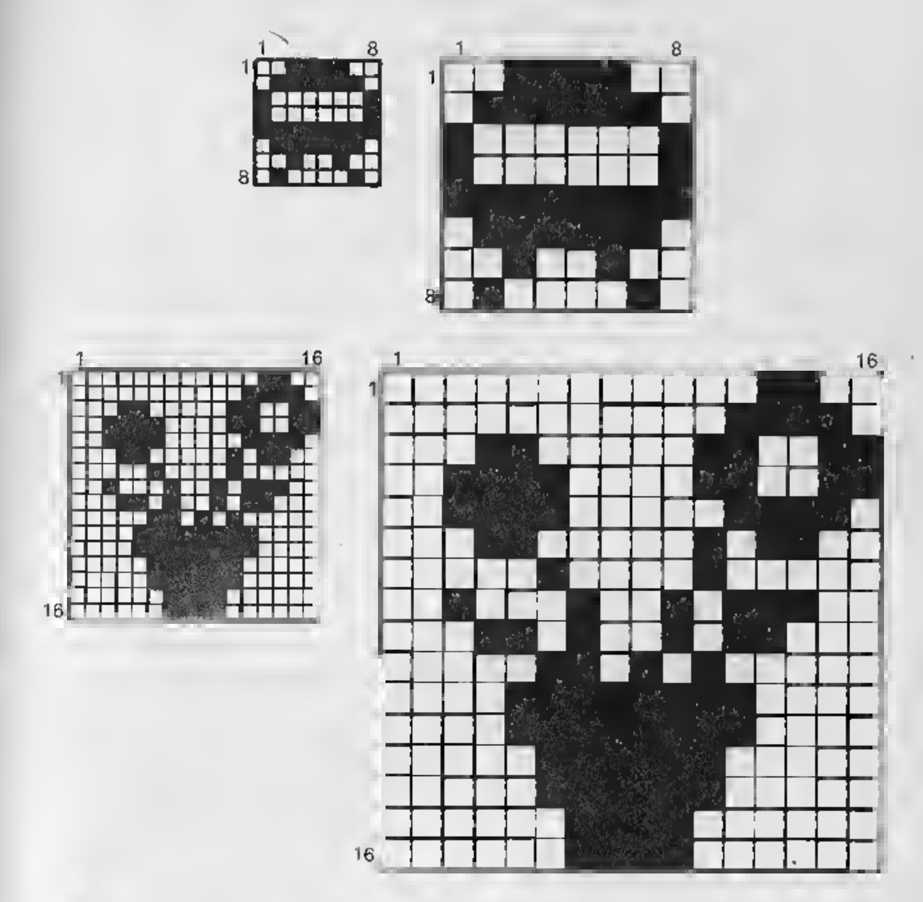
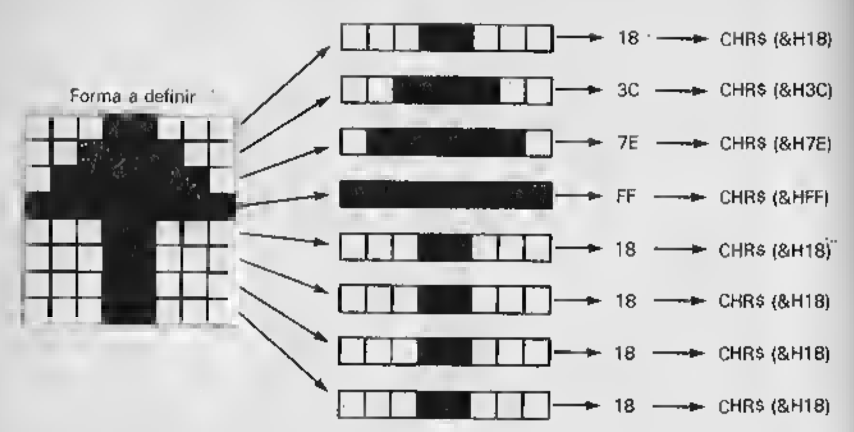


Fig. 3 - Sprites de 8×8 y 16×16 en sus dos dimensiones.

Damos a continuación la tabla de conversión de las posibles líneas individuales de los sprites a hexadecimal, que puede resultar muy cómoda para los que no manejen con soltura las conversiones desde el sistema de numeración binaria.

El color de un sprite, el plano y la posición en que deseamos que aparezca se definen con la instrucción PUT SPRITE. Son posibles 256 sprites de 8×8 (de Ø a 255) y 64 sprites de 16×16 (de Ø a 63).

En un plano se puede visualizar un solo sprite a la vez. Los sprites de planos anteriores (con número menor) borran total o parcialmente los de planos posteriores en caso de superposición.



SPRITE$ (1)=CHR$ (&H18)+CHR$ (&H3C)+CHR$ (&H7E)+CHR$ (&HFF)+CHR$ (&H18)+CHR$ (&H18)+CHR$ (&H18)+CHR$ (&H18)


Fig. 4 · Construcción de un sprite.

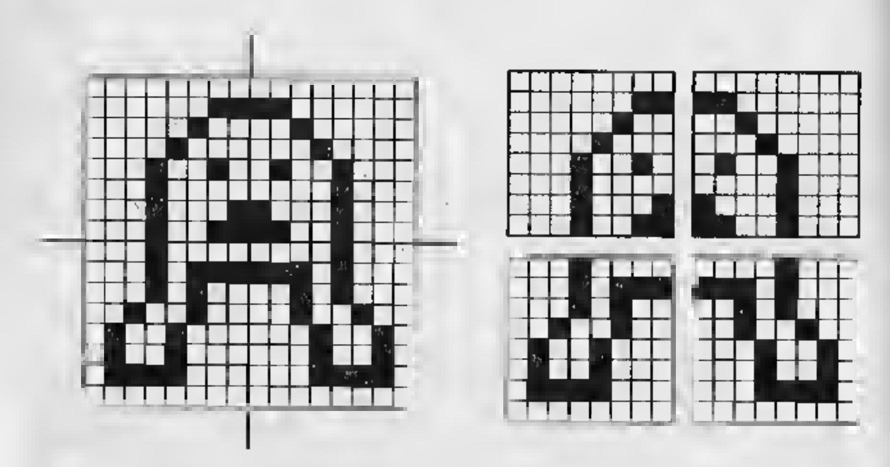


Fig. 5 - Dos ejemplos de sprites de 16×16.

Se pueden visualizar como máximo cuatro sprites a la vez; si intentamos visualizar más sólo veremos los cuatro de los planos con números más bajos.

El movimiento de los sprites se realiza siempre con PUT SPRITE, de hecho, cada ejecución de PUT SPRITE hace que desaparezca el sprite visualizado hasta ese momento y que

Forma	Hexadecimal	Forma	Hexadecimal
□□□□	0	■□□□	8
□□□■	1	■□□■	9
□□■□	2	■□■□	A
□□■■	3	■□■■	B
□■□□	4	■■□□	C
□■□■	5	■■□■	D
□■■□	6	■■■□	E
□■■■	7	■■■■	F

Fig. 6 · Cuadros de sprites expresados en hexadecimal.

aparezca otro nuevo; si las coordenadas son distintas lograremos una sensación de movimiento.

Pasemos ya a estudiar las instrucciones gráficas

Instrucciones gráficas

SCREEN [modo] [, dimensión del sprite] [, activación sonido teclas] [, velocidad casete] [, tipo de impresora].

Define las características de visualización, la presencia del sonido que indica pulsación de una tecla, la velocidad de transmisión a la grabadora en baudios y el tipo de impresora si no se ajusta al estándar MSX. Veamos cada uno de estos puntos (los valores de defecto en cada uno están identificados por el color azul):

MODO

Ø: modo texto, 40 car×24 líneas; cada carácter tiene una anchura de 6 puntos horizontales; no se puede usar el plano del sprite ni las instrucciones gráficas.
1: modo texto, 32 car×24 líneas; cada carácter tiene una anchura de 8 caracteres horizontales; se pueden usar los planos de los sprites, pero no las instrucciones gráficas.
2: modo gráfico de alta resolución; la pantalla tiene 256 puntos horizontales por 192 verticales; los gráficos se trazan por puntos; se pueden usar los planos de los sprites.
3: modo gráfico de baja resolución, multicolor, como el modo 2, pero los gráficos se trazan por bloques de 4×4 puntos.

DIMENSION DEL SPRITE

Ø 1×8 no ampliados
1 8×8 ampliados
2 16×16 no ampliados
3 16×16 ampliados

En un mismo programa se pueden definir varias formas de sprites con la instrucción SPRITE(n), que define la forma del sprite n-ésimo.

ACTIVACION SONIDO TECLAS

Ø no hay sonido al pulsar una tecla (excepto su "click", claro)
1-255 si lo hay

VELOCIDAD CASETE

La transmisión de datos entre ordenador y el casete depende de la interface usada; con este parámetro podemos establecer la velocidad de transmisión en baudios (bits/seg).

1 1200 baudios
2 2400 baudios

TIPO DE IMPRESORA

Ø MSX
1-255 distinta de MSX

WIDTH (número de caracteres)

Define la capacidad de una línea horizontal de la pantalla en número de caracteres. Sólo se puede usar en modo texto.

CLS

Borra todo lo visualizado en la pantalla.

LOCATE [x] [, y] [, cursor]

Mueve el cursor al punto de coordenadas (x, y), visualizándolo o no según el valor del parámetro "cursor" sea 1 ó Ø. Se puede emplear, por tanto, para situarnos en el punto donde queremos empezar a dibujar o escribir.
El valor de defecto para las coordenadas es Ø, y para el cursor, 1. Es válido en todos los modos de pantalla.

COLOR (primer plano), (fondo), (bordes)

Define el color del primer plano, del fondo y de los bordes de la pantalla.
Los parámetros especificados tienen que asumir valores comprendidos entre Ø y 15; si no son enteros se truncan.
Para cambiar de color en modo gráfico hay que ejecutar un CLS después de COLOR.
La tabla siguiente establece la relación entre cada código y su color asociado.

CODIGO DE LOS COLORES			
0	transparente	9	rojo semi-oscuro
1	negro	10	amarillo oscuro
2	verde semi-oscuro	11	amarillo claro
3	verde claro	12	verde oscuro
4	azul marino	13	magenta
5	azul claro	14	gris
6	rojo oscuro	15	blanco
7	azul		
8	rojo semi-oscuro		


Tabla 1 - Código de los colores.

PUT SPRITE plano [, (x, y)] [, color] [, sprite]

Hace aparecer el sprite dado en la posición (x, y) del plano de sprites determinado.
Las coordenadas x, y localizan el punto superior izquierdo del sprite (que es el usado para posicionar el sprite) y pueden ser

cualquier expresión numérica cuyo valor esté entre -32 y 255 para "x", y entre -32 y 191 para "y". Los valores negativos sirven para hacer desaparecer parcialmente el sprite cuando está entrando o saliendo de la pantalla.

La coordenada "y" puede asumir también dos valores convencionales, 208 y 209; en el primer caso desaparecen todos los sprites de los planos posteriores al especificado, mientras que en el segundo desaparece el sprite en cuestión.

El "plano" tiene que tener un valor entero comprendido entre Ø y 31.

El "color" tiene que ser un entero entre Ø y 15; si no es especificado se adopta el color actual del primer plano.

El "sprite" debe ser un número entero comprendido entre Ø y 255 si el sprite es de 8×8, y entre Ø y 63 si el sprite es de 16×16; si no se concreta, el ordenador toma el mismo número del plano.

Los valores del plano, del color y del sprite se pueden dar con cualquier expresión; si no son enteros se truncan.

Esta instrucción se puede usar en el modo 1 (de texto) y en ambos modos gráficos 2 y 3.

SPRITE ON

Activa la detección de colisiones entre sprites. Si se producen pone la variable SPRITE a 1, y si no, en Ø.

SPRITE OFF

Desactiva la detección de colisiones entre sprites.

CIRCLE (x, y), radio [, color] [, ángulo de partida] [, ángulo de llegada] [, ovalidad]

Se puede usar sólo en modo gráfico. Traza una línea curva en primer plano, con centro en (x, y), radio especificado, y extensión desde el ángulo de partida al de llegada; la ovalidad sirve para representar una curva más o menos cercana al círculo.

Las coordenadas x, y son expresiones numéricas cuyo valor está entre -32768 y 32767, así como el radio.

El código "color" es un número entero comprendido entre Ø y 15; si no se especifica se adopta el color actual del primer plano.

El ángulo de partida y el de llegada se expresan en radiales, y pueden variar entre -2π y $+2\pi$; el valor de defecto del primero es Ø, y del segundo, $+2\pi$.

La ovalidad puede ser una expresión numérica cualquiera; el valor de defecto es 1, que corresponde a una elipse; el valor necesario para un círculo perfecto es 1,4.

DRAW subcomandos

Se usa sólo en modo gráfico. Traza en la pantalla un dibujo cuyas características determinan los subcomandos especificados.

Los subcomandos no son más que cadenas de caracteres especiales. Sus tipos y significados son los siguientes:

—SUBCOMANDOS

Sn

Ø⩽n⩽255

Especifica el factor de escala, es decir, el factor por el que se multiplican las coordenadas en los subcomandos U, D, R, L, E, F, G, H, M, que será n/4. El valor de defecto es S4, que provoca un factor de 1.

AN

Ø⩽n⩽3

Hace girar el sistema de coordenadas un ángulo igual a n×90, a partir de una posición inicial correspondiente a AØ.

Las cuatro configuraciones posibles son las indicadas en la figura 7.

Cn

Ø⩽n⩽15

Especifica el color del gráfico; el valor de defecto es 15 (blanco).

M x, y

Ø⩽x⩽255
Ø⩽y⩽191

Traza una línea desde el punto actual al de coordenadas (x, y), siendo estos valores absolutos respecto al sistema de coordenadas vigente en el momento.

M±x, ±y

Ø⩽x⩽255
Ø⩽y⩽191

Como el anterior, pero las coordenadas se dan en valor relativo al punto actual. Los signos representan los desplazamientos en el sentido positivo o negativo de los ejes del sistema en uso.

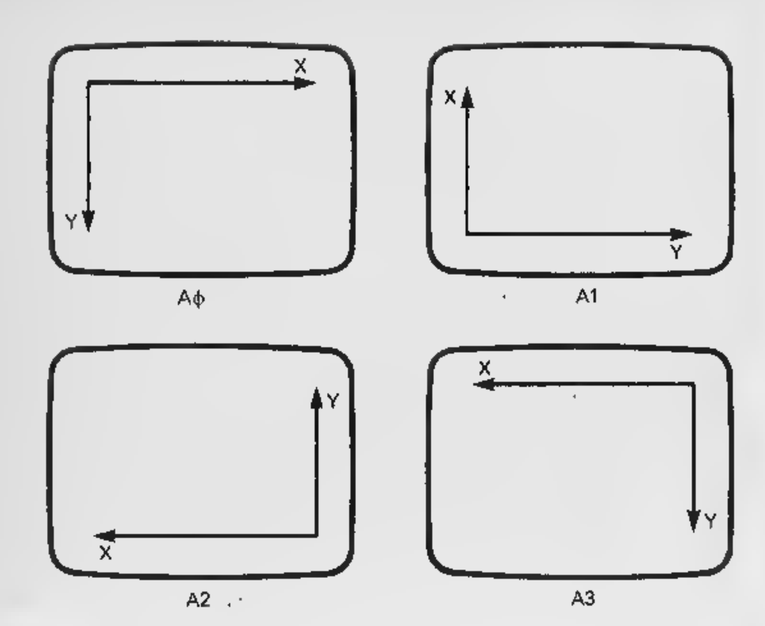


Fig. 7 - Sistemas de coordenadas.

La figura 8 muestra un ejemplo (ver lo referente a prefijos de esta misma instrucción).

Un

Ø⩽n

Traza un segmento de recta paralelo al eje y, en sentido negativo, a partir del punto actual, hasta un punto distante n; el valor de defecto de n es 1. Es, por tanto, un movimiento hacia arriba (Up).

Dn

Como el anterior, pero en el sentido del eje y creciente; movimiento hacia abajo (Down).

Rn

Ø⩽n

Traza un segmento de longitud n paralelo al eje x, en el sentido de las "x" crecientes. Movimiento hacia la derecha (Right).

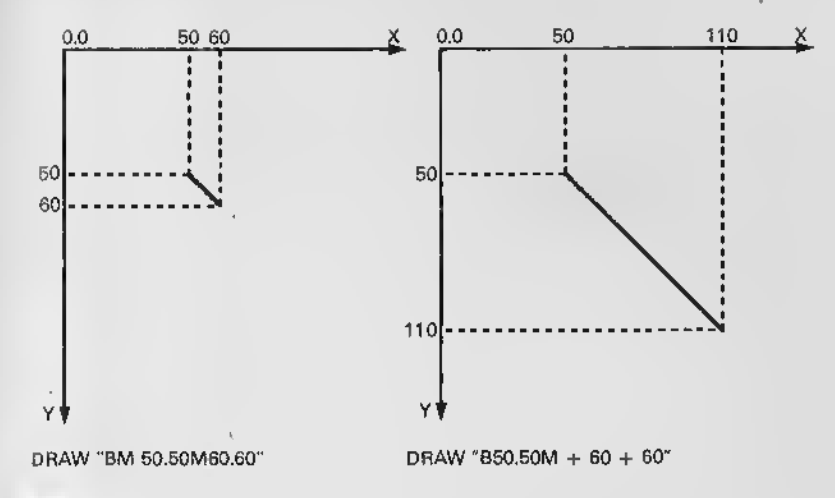



Fig. 8 - Ejemplos de uso de DRAW.

Ln

Como el anterior, pero en el sentido negativo del eje x. Movimiento hacia la izquierda (Left).

En

Ø⩽n

Traza un segmento desde el punto actual (x, y) a un punto de coordenadas (x+n, y–n). Movimiento diagonal arriba-derecha.

Fn

Como el anterior, desde el punto actual al punto (x+n, y+n). Movimiento diagonal abajo-derecha.

Gn

Como los anteriores, desde el punto actual hasta el punto (x–n, y+n). Movimiento diagonal abajo-izquierda.

Hn

Como los anteriores, desde el punto actual al (x–n, y–n). Movimiento diagonal arriba-izquierda.

PREFIJOS

B Cuando un subcomando está precedido por B, el punto actual se desplaza conforme aquél pero no se dibuja la línea.

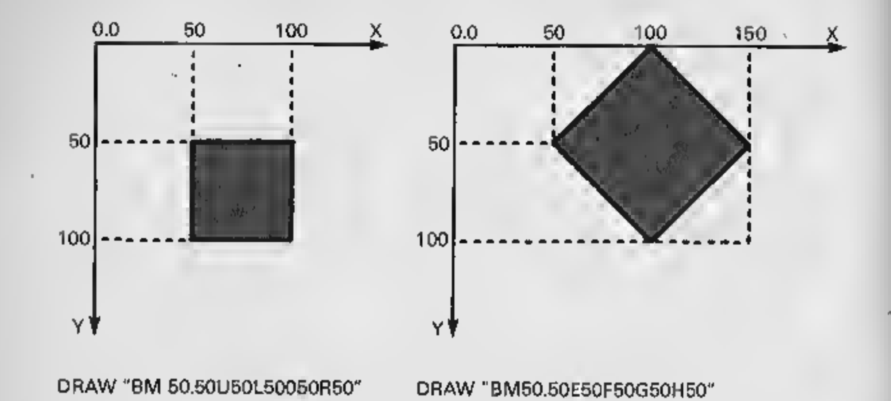


Fig. 9 - Ejemplos de empleo de DRAW.

N Cuando un subcomando está precedido por N se dibuja la línea pero no se desplaza el punto corriente.

X Un subcomando se puede expresar como constante de cadena, entre comillas, o como una variable de cadena o como una expresión. Una parte de un subcomando, común a más de uno, se puede representar con una variable de cadena e insertarla de esta forma en los subcomandos; en éstos tiene que estar precedida por la letra X y seguida por un punto y coma (;).

= El parámetro n de los subcomandos puede ser una constante numérica o una variable; en este caso tiene que estar precedido por el signo = y seguido de un punto y coma (;).

LINE [(xa, ya)–] (xb, yb) [, código color] [, B o BF]

Traza un segmento desde el punto "a" al punto "b", con el color especificado.

Si se especifica B traza un rectángulo con el segmento como diagonal, si se especifica BF colorea la superficie delimitada por el rectángulo.

El punto a se puede omitir y, en ese caso, se asume como punto inicial el actual.

El color de defecto es el actual del primer plano.

Las coordenadas pueden ser cualquier expresión numérica cuyo valor esté entre –32768 y 32767.

Para obtener cuadrados, el segmento deberá ser 1,4 veces superior al valor del lado deseado.

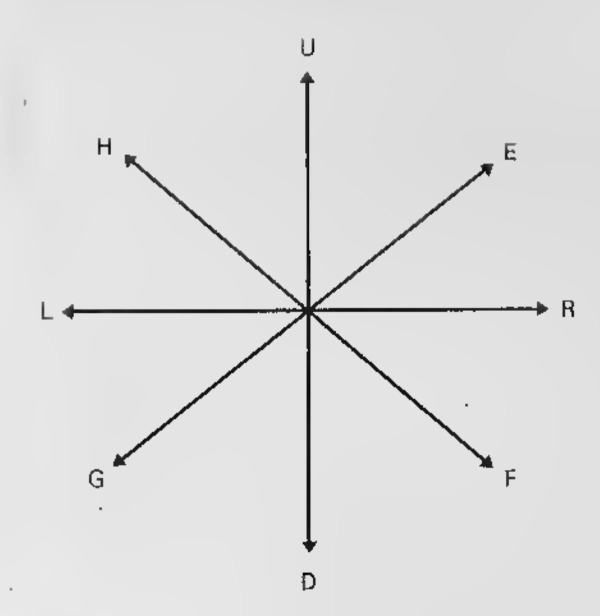


Fig. 10 - Resumen "gráfico" de los subcomandos para desplazamiento que admite la instrucción DRAW.

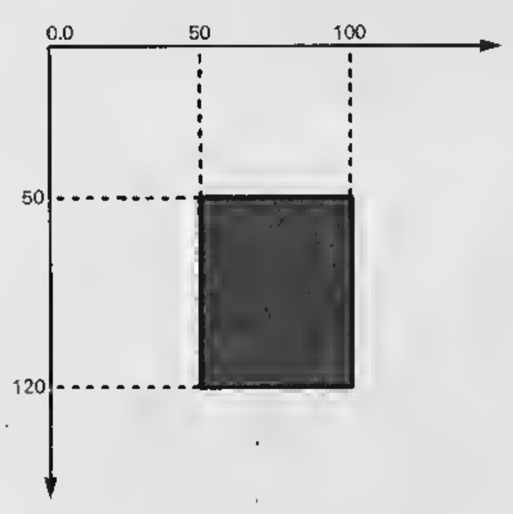


Fig. 11 - Ejemplo de LINE.

PAINT (x, y) [, color interno] [, color del borde]

Colorea la superficie en cuyo interior está el punto (x, y) y delimitada por una línea de contorno; si el contorno no está cerrado colorea toda la pantalla.

Las coordenadas "x" e "y" son expresiones numéricas cuyo valor está entre Ø y 255 para "x", y entre Ø y 191 para "y".

Los códigos de los colores tienen que estar entre Ø y 15; si se omiten se sustituyen por el color del primer plano.

En modo SCREEN 2 se tiene que dar el mismo color al interior y al contorno de la figura; de otra forma se colorearía toda la pantalla. En modo SCREEN 3 es posible utilizar colores distintos.

POINT (x, y)

Retorna el código del color existente en el punto (x, y).

PSET (x, y) [, color]

Se usa en los modos gráficos, dibuja el pixel de coordenadas (x, y) con el color especificado. El color de defecto es el del primer plano.

PRESET (x, y) [, color]

Se usa en los modos gráficos; es similar a PSET, pero el color de defecto es el del fondo. De este modo se puede "borrar" un punto.

KEY OFF

Elimina la visualización de las cadenas de caracteres asociadas a las teclas de función, que normalmente aparecen en la última línea de la pantalla.

KEY ON

Restablece la visualización en la última línea de la pantalla de las cadenas de caracteres asociadas a las teclas de función.



CAPITULO V

SONIDO MSX

Las posibilidades sonoras del MSX son, sin lugar a duda, dignas de elogio, con resultados realmente sorprendentes.

Con dos comandos (PLAY y SOUND) y con una gran variedad de subcomandos que luego veremos en detalle, se obtienen sonidos de todos los tipos, llegando incluso a imitar distintos instrumentos musicales.

El sistema MSX utiliza un PSG (Generador de Sonidos Programable), capaz de producir tres sonidos simultáneamente. Las características de los sonidos generados están determinadas por los valores contenidos en 16 registros de dicho procesador, de los cuales 13 son programables directamente en BASIC con la instrucción SOUND.

La instrucción PLAY obtiene resultados similares a través de subcomandos, que tienen que ser luego interpretados por el sistema y enviados entonces a los mismos registros.

El PSG de los equipos MSX es el AY-3-891Ø de General Instruments. Sus 13 registros accesibles son:

n.º registro	bits usados	Función
Ø	8	Ajuste fino de tono del canal A (Ø-255).
1	4	Ajuste grueso de tono del canal A (Ø-15).
2	8	Ajuste fino de tono del canal B (Ø-255).

3	4	Ajuste grueso de tono del canal B (Ø-15).
4	8	Ajuste fino de tono del canal C (Ø-255).
5	4	Ajuste grueso de tono del canal C (Ø-15).
6	5	Canal de ruido (Ø-31).
7	6	Cambia la salida de un tono (valores inferiores a 8) a un ruido (superiores a 7) en los canales.
8	5	Volumen del canal A.
9	5	Volumen del canal B.
1Ø	5	Volumen del canal C.
11	8	Duración.
12	8	Duración.
13	4	Controlador de envolvente.

Vayamos ahora con las tres instrucciones posibles: PLAY, SOUND y BEEP.

PLAY subcomandos

Genera uno o más sonidos, cuyas características están especificadas por los subcomandos que usemos.

—SUBCOMANDOS

Tn

32≤n≤255

Determina la velocidad de la música en términos de cadencia de cuartos de notas por ninuto; la velocidad de ejecución varía de forma lineal con n.

El valor de defecto es T12Ø.

On

1≤n≤8

Determina la octava en la que se van a tocar las notas especificadas a continuación con las letras A-G; la octava crece al crecer n de 1 a 8; la octava correspondiente a O4 es la de la figura 1.

El valor de defecto es precisamente O4.

Ln

1≤n≤64

Determina la duración del sonido de las notas siguientes. Su longitud será 1/m, de forma que L1 equivale a una redonda, L2 a

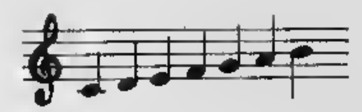



Fig. 1 - Octava O4.

una blanca y así sucesivamente hasta L64 que es la semifusa, como se ve en la figura 2.

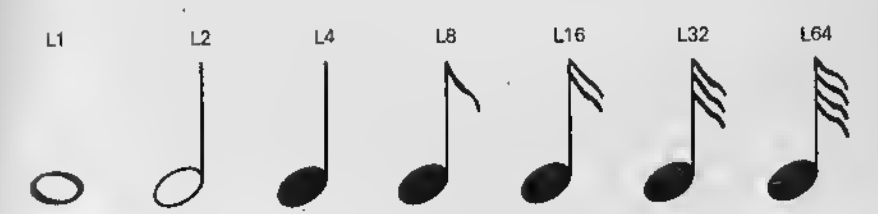



Fig. 2 - Duración de las notas.

Nn

Ø≤n≤96

Especifica una nota musical independiente de la octava seleccionada con el comando O; NØ representa un silencio (cuya duración depende del comando L) 1 la nota C de la octava más baja, y así sucesivamente. N36 corresponderá a la nota de la figura 3.

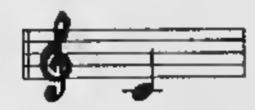



Fig. 3 - Nota N36.

A-G
An-Gn

1≤n≤64

Ejecuta la nota indicada dentro de la octava definida con el subcomando O. Se pueden usar los caracteres # y + para indicar

un sostenido y para los bemoles. Sin embargo, esto sólo es válido si se corresponde con una tecla negra del piano.

La correspondencia entre las letras usadas por americanos e ingleses para denominar las notas y las palabras europeas son:

A=LA
B=SI
C=DO
D=RE
E=MI
F=FA
G=SOL

si queremos especificar la duración concreta de una nota; en el caso de que sea distinta de la especificada con el subcomando L, podemos usar la "n". La figura 4 aclara el significado de las distintas notas con eventuales semitonos.

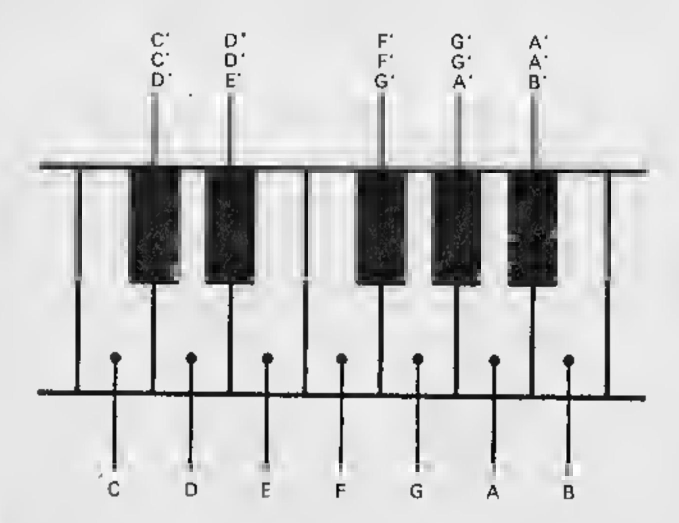


Fig. 4 · Notas y semitonos dentro de una octava.

Rn

1⩽n⩽64

Determina la duración de un intervalo de silencio; funciona como L. El valor por defecto es 4. La figura 5 muestra varias duraciones posibles.




Fig 5 · Duración de los silencios.

Después de una nota o de un intervalo aumenta 3/2 su duración, es decir, la ejecuta con puntillo. Por eso puede ponerse también como uno o más puntos (su número equivale a "n").

Por ejemplo, A.. hace que suene el LA con 9/4 su duración.

Vn

Ø⩽n⩽15

Determina el volumen del sonido; n=Ø no produce sonido. El valor de defecto es V8.

Sn

Ø⩽n⩽15

Determina la variación del volumen en base a una de las distintas curvas representadas en la figura 6:

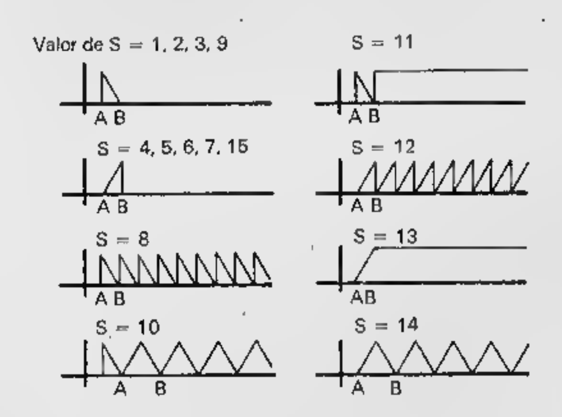



Fig. 6 - Distintas "envolventes" de las notas según el valor dado con S.

La duración del segmento A-B para cada uno de los gráficos anteriores está determinada por el comando M.

Los frentes de subida o de bajada verticales significan el paso "instantáneo" del silencio al máximo nivel y viceversa; los frentes de subida y de bajada oblicuos representan aumentos y disminuciones graduales de volumen.

El valor de defecto es S1.

Para entender los efectos del subcomando S le aconsejamos intentar generar una sola nota con la máxima duración, aplicando luego los distintos valores de S, con un programa como el guiente:

```
10 A$="T120O4L1AVBS"
20 INPUT B$
30 C$=A$+B$
40 PLAY C$
```

Mn

1⩽n⩽65535

Determina la duración del segmento A-B del comando S (gráficos de la figura 6). La duración aumenta de forma lineal según varía n.

El valor por defecto es M255.

La duración del segmento A-B puede variar desde 0.0001 segundos, correspondiente a M1, a 10 segundos, correspondiente a M65535; la razón de estos números se explica en la descripción de la instrucción SOUND.

NOTA: Los valores por defecto se adoptan sólo la primera vez que se utiliza una instrucción PLAY en el programa. Si no se especifica de otra forma en las demás instrucciones PLAY se mantienen los valores asignados con el último subcomando.

Evidentemente, si en un programa no usamos un subcomando dado, todos los PLAY tomarán su valor por defecto, *ya que es el que se asignó en el primer PLAY.*

Le sugerimos un programa muy sencillo que le permitirá ensayar las distintas combinaciones del subcomando M (o de otros cualquiera que desee):

```
10 INPUT A$
20 B$=A$+C$
30 PLAY B$
40 GOTO 10
```

Intente con los datos siguientes (pulse STOP para interrumpir el programa):


```
t3212s1m160a
t3212s1m1600a
t3212s1m16000a
t3212s1m32500a
t3212s1m65535a
```

Los resultados se representan gráficamente en la figura 7.

La misma secuencia de datos, con 98 en lugar de 91, da, en cambio, los resultados de figura 8.

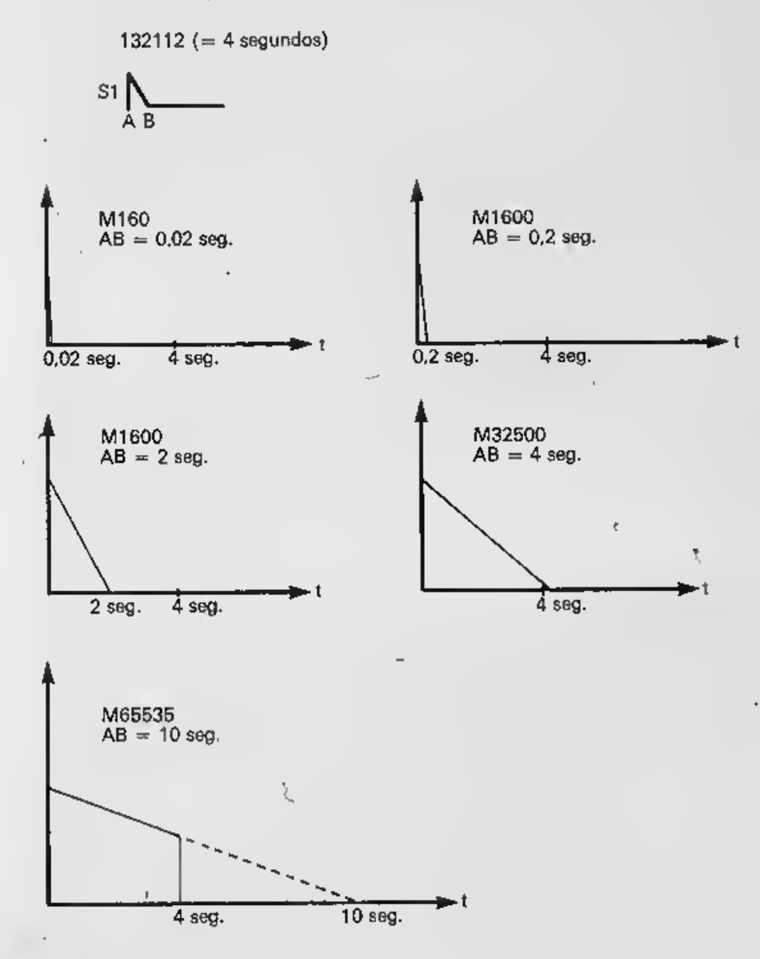


Fig. 7 - Representación gráfica de los sonidos que podemos obtener con S1.

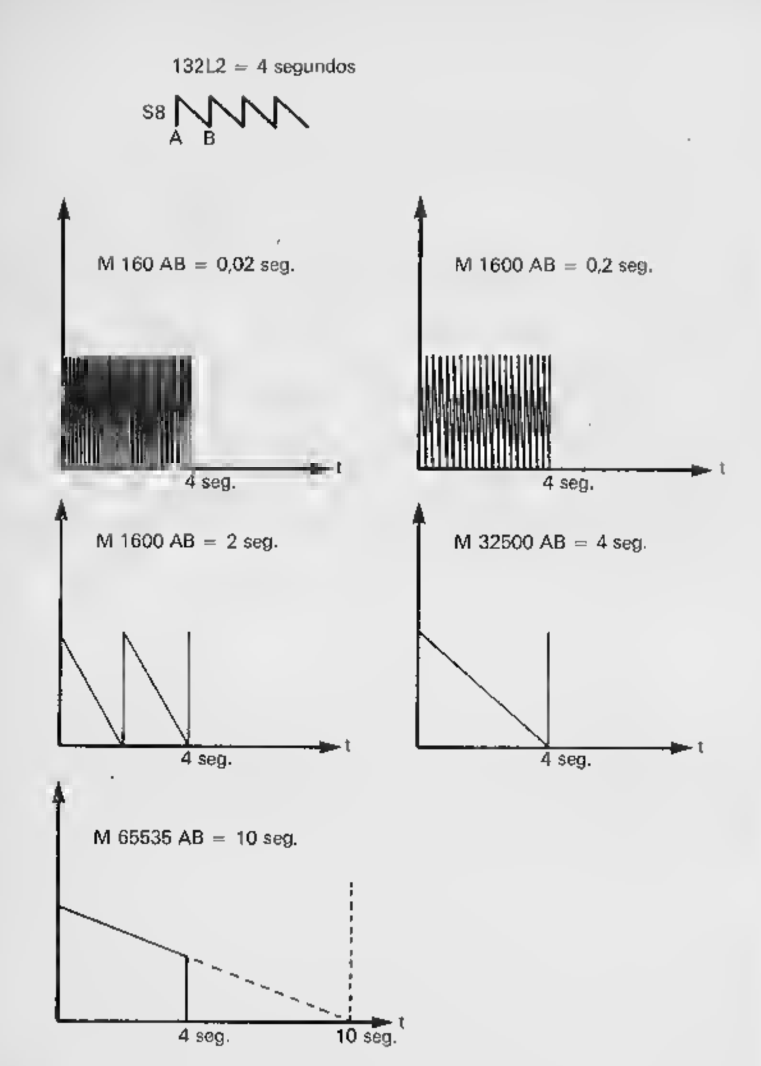


Fig. 8 - Representación gráfica de los sonidos que podemos obtener con S8.

Como habrá podido comprobar, el funcionamiento de los subcomandos es bastante complejo. Centrémonos ahora en el proceso de "creación musical", aprovechando de paso para repasar conceptos.

Consideremos una única nota, por simplicidad. Con el subcomando L se fija su duración, es decir, cuánto tiempo dura la emisión del sonido. Con el 9 se fija la forma de variación del volumen, y con el M, la longitud del segmento A-B, que, básicamente, es el lapso de tiempo en el que el volumen pasa de Ø al valor máximo y vuelve a Ø.

Supongamos que hemos fijado ya la duración de una nota y que ponemos S1; todavía podemos tener distintos resultados jugando con el valor dado al subcomando M. Con un M intermedio oirá la nota inmediatamente al volumen máximo y luego, como se va apagando gradualmente; con un M muy pequeño la duración del pico A-B es tan breve que no alcanzará a oír la nota, cuyo volumen baja inmediatamente a Ø; por el contrario, con un M muy grande tendrá la impresión de que la nota permanece al mismo volumen a lo largo de todo el tiempo, ya que su disminución es extremadamente lenta.

Si usamos S8 en lugar de S1, cuanto más pequeño sea M tantas más veces oirá repetir la nota en toda su duración, aunque si M es demasiado pequeño se repetirá el fenómeno anteriormente explicado y no oirá nada. Cuanto mayor sea M tanto más nítidamente oirá la nota, pero también menos veces.

Si la duración de la nota es muy breve y el valor asignado a M es más bien grande, puede darse el caso de que algunas configuraciones con valores distintos de S den el mismo resultado, porque la parte reproducida de la nota cae en zonas iguales de gráficos distintos.

Ejecute el último programa que vimos e introduzca los datos:

```
t3212s1m40000
t3212s8m40000
```

El resultado se representa en la figura 9.

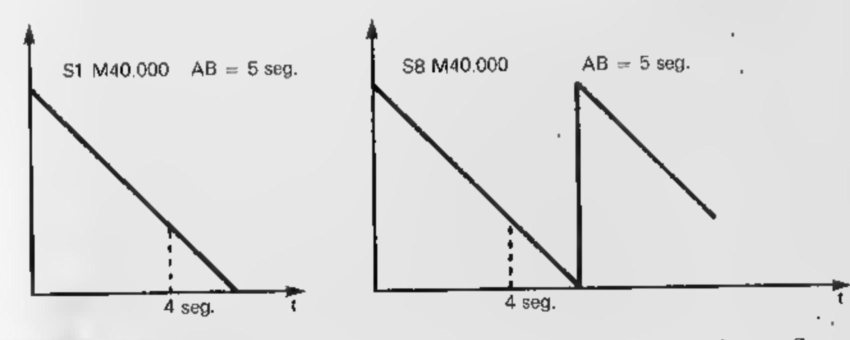


Fig. 9 - Resultados iguales, al ser M grande, aunque variemos S.

Con cierta experiencia y comparando muchos casos distintos podrá determinar la forma de coordinar entre sí los valores de los subcomandos L,S y M para obtener música y no ruidos.

NOTA: En una cadena de subcomandos se puede incluir una variable de cadena a la que se hayan asignado anteriormente uno o más subcomandos; la variable en cuestión tiene que estar precedida por X y seguida por punto y coma (;) en la cadena.
El valor n de un subcomando puede estar expresado con una variable numérica, que en este caso tiene que estar precedida por = y seguida por ;.

SOUND número de registro, expresión

Como hemos dicho, el sistema MSX utiliza un chip para generar los sonidos, dotado de 16 registros programables; 13 de éstos son accesibles directamente desde BASIC por el usuario mediante la instrucción SOUND, que escribe en el registro cuyo número se especifica como primer parámetro el valor obtenido de la expresión que aparece como segundo parámetro.
El chip tiene tres canales, en cada uno de los cuales se puede producir un sonido de frecuencia distinta y/o ruido (que tiene que tener la misma frecuencia en los tres canales). Además, en cada canal se puede fijar el volumen máximo y una variación del volumen idéntica a la que se obtiene con los comandos S y M de la instrucción PLAY.
Veamos ahora el significado y la función de los distintos registros.

Registros Ø-1, 2-3, 4-5

Establecen la frecuencia del sonido emitido, respectivamente, por los canales A, B y C. Para obtener una frecuencia de valor "f" hay que introducir en la pareja de registros el valor N dado por:

f = 1996750/(16×N)   N = 1996750/(16×f)

Este valor calculado se convierte primero a binario y luego se subdivide en un byte bajo, cuyo contenido puede variar desde Ø hasta 255, y en un byte alto, cuyo contenido puede variar de Ø a 15. El byte bajo se escribe en el registro con número par de la pareja relativa al canal elegido, y el byte alto, en el registro con número impar.
Las fórmulas que nos dan los valores que hay que introducir en los dos registros (alto=High, bajo=Low) en función de la frecuencia "f" son:

H = N/256
L = N-INT(N/256)×256

El mínimo valor distinto de Ø representable en la pareja de registros es, naturalmente, el 1, que nos da el valor máximo de la frecuencia, igual a:

$f_{máx.}$ = 1996750/(16×1)=124000 Hz aprox.

En cambio, el máximo valor representable en los 12 bits es 4095 (255+15×256=2^{12}−1), al que corresponde el valor mínimo de frecuencia, igual a:

$f_{mín.}$ = 1996750/(16×4095)=30 Hz aprox.

La situación está esquematizada en la figura 10.

Registro 6

Determina la frecuencia del ruido, con una fórmula idéntica a la que hemos visto para el sonido.
Su contenido puede ser como máximo 31 (disponemos de 5 bits), lo que significa que la frecuencia de ruido puede variar desde:

1996750/(16×31)=4026 Hz aprox.

hasta cerca de 124000 Hz, como hemos visto para el sonido.

Registro 7

En este registro se utilizan sólo los 6 bits menos significativos; cada uno de ellos activa (si está a Ø) o desactiva (si está a 1) el

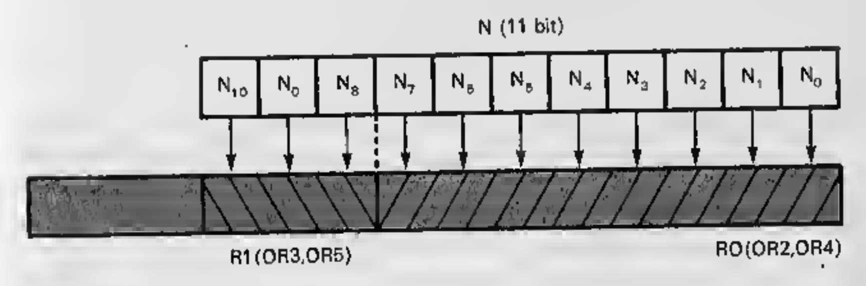
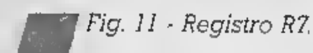


Fig. 10 - Registros RØ y R1 (o bien R2-R3, R4-R5).




Fig. 11 - Registro R7.

sonido o el ruido en uno de los tres canales, como se ve en la figura 11.

La configuración con todos los bits a 1 (63 decimal, 3F hexadecimal) no genera señal sonora alguna en ninguno de los tres canales; para generar sonido o ruido en uno o más canales hay que poner a Ø los bits correspondientes.

La configuración de la figura 12, por ejemplo, produce sonido en los canales A y B y ruido en el canal C; para obtenerla es suficiente asignar al registro 7 el valor 28, que se obtiene restando a 63 la suma de los valores de los bits que hay que poner a Ø:

28=63-(1+2+32)

Registros 8, 9, 1Ø

Establecen el volumen de los canales A, B y C, respectivamente. Pueden contener un valor entre Ø y 16. Si usamos uno entre Ø y 15 actúa de la misma forma que el subcomando V en la instrucción PLAY. El valor 16 sirve para producir en el canal seleccionado la variación del volumen de la forma especificada en el registro 13, como veremos más adelante.

Registros 11, 12

Estos dos registros tienen la misma función que el subcomando M en la instrucción PLAY, al igual que el registro 13 tiene la

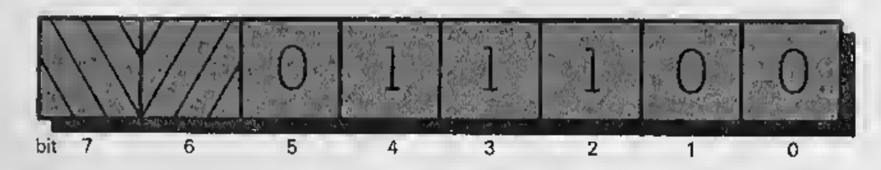


Fig. 12 - Valores necesarios para lograr sonido en los canales A y b, y ruido en el C.

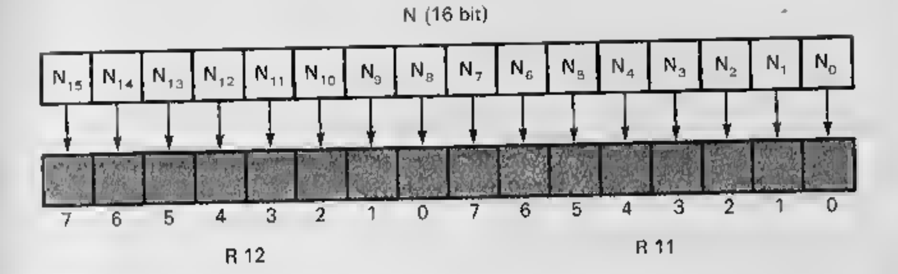


Fig. 13 - Registros 11 y 12.

función del subcomando S. Contienen un valor N conjunto comprendido entre 1 y 65535, subdividido en byte bajo y byte alto y cargados, respectivamente, en los registros 11 y 12. Producen un segmento A-B de duración:

duración (seg.) = 256×N/199675Ø

El valor 1 corresponde a una duración del segmento A-B igual a:

256*1/199675Ø=Ø.ØØØ1 seg. aprox.

El valor 65535 corresponde a una duración del segmento A-B igual a:

256*65535/199675Ø=1Ø seg. aprox.

La situación de los dos registros está esquematizada en la figura 13

Registro 13

Determina la variación del volumen en el canal seleccionado, con un número comprendido entre Ø y 14. Cada valor tiene el mismo significado visto cuando se explicó el subcomando S de la instrucción PLAY.

BEEP

Genera una señal acústica fija de corta duración.

CAPITULO VI

FICHEROS DE PROGRAMAS Y DE DATOS

El tratamiento de los ficheros es uno de los temas más importantes y complejos de la programación. Por otra parte, en cuanto se abandona el terreno de las aplicaciones elementales, que sin duda no son las que justifican la necesidad de un ordenador, se hace indispensable saberse mover con soltura entre casetes y disquetes (floppy-disck).

El ordenador maneja y elabora la información contenida en la memoria central, que, como sabemos, puede ser tanto las instrucciones del programa en ejecución como los datos sobre los que trabaja dicho programa. La memoria central tiene, además de toda una serie de ventajas extraordinarias, dos graves inconvenientes: su limitada capacidad (ligada a cuestiones de tipo tecnológico que no merece la pena profundizar aquí) y, sobre todo, que pierde su contenido cuando se apaga el ordenador.

Por cortos y sencillos que sean los programas de un principiante, éste se dará pronto cuenta de lo molesto que resulta tener que teclearlos cada vez que vuelve a encender el ordenador y desea ejecutarlos. ¡Imagínese entonces lo que supone esta situación con un programa largo y complejo!

El mismo problema se presenta, aunque probablemente no para quienes estén comenzando a ejercitarse, por lo que se refiere a los datos: supongamos, por ejemplo, que un programa tenga que calcular cada mes los sueldos de una empresa de 1.000 empleados; evidentemente el problema ya no se reduce sólo a conservar el programa para ejecutarlo cada mes, sino que también habrá que guardar toda la información relativa a los empleados o, por lo menos, la que permanece constante en el cálculo del suel-

do. Si no hiciéramos esto todos los meses habría que realizar un trabajo inmenso para introducir los datos por el teclado, lo cual, aunque por una parte no anula, sí reduce mucho la ventaja que se deriva del uso de un ordenador.

Las memorias de masa (o de almacenamiento masivo) tienen justamente la finalidad de conservar tanto programas como datos indefinidamente, en forma de bloques o conjuntos de información a los que se les llama ficheros. Las características más notables de estas memorias, que suelen ser cintas o discos magnéticos, son: mayor capacidad y menor coste con relación a la memoria central y, sobre todo, la posibilidad de memorizar información de forma permanente.

Dado que cualquier tipo de información para ser elaborada por la CPU tiene que estar disponible en la memoria central, la utilización de grandes ficheros guardados en memorias de masa requiere continuas transferencias de información entre los dos tipos de memorias.

El dispositivo de memoria de masa más versátil y eficiente para los ordenadores personales es la unidad de disco, aunque



Fig. 1 - Mediante un simple casete de audio tenemos acceso a la gran biblioteca de programas MSX y podemos conservar nuestros ficheros de programas y datos.

también es bastante más cara y un poco más compleja de usar que las grabadoras de casete, con lo que no siempre resulta fácilmente accesible al usuario. Para permitirnos su uso está el MSX-DOS (Disk Operating System, sistema operativo disco). Nosotros nos centraremos en cómo utilizar el casete, no sólo por lo sencillo de su uso y por su precio más asequible (además de que sirve cualquier casete de audio que tengamos), sino especialmente por su mucha mayor difusión.

Podemos dividir las instrucciones para la gestión de los ficheros en dos grupos: las correspondientes a ficheros de programas y las de ficheros de datos.

Extendiendo el concepto de fichero más allá de las memorias de masa, podemos considerar también la impresora y la pantalla como dispositivos sobre los cuales se pueden crear ficheros; en el primer caso el fichero se graba sobre papel, es decir, se imprime, mientras en el segundo se memoriza en la pantalla o, lo que es lo mismo, se visualiza. Sin embargo, si bien es posible escribir un fichero en uno de estos dispositivos, no podremos leerlo.

Ficheros de programas

Un programa BASIC se puede salvar en casete tanto en formato ASCII (mediante SAVE) como binario (con CSAVE), y se puede luego cargar en memoria, respectivamente, con LOAD y CLOAD. Los programas cargados con LOAD son los únicos que se pueden visualizar en la pantalla. Un programa almacenado en casete se puede fusionar con el que tengamos en la memoria central con un comando MERGE, siempre que haya sido grabado en formato ASCII con un comando SAVE. Veamos los distintos comandos que nos facilitan el manejo de los ficheros de programas en casete.

SAVE "nombre dispositivo:[nombre fichero]"

Graba el programa que se encuentra en memoria en el dispositivo seleccionado (cinta, pantalla o impresora), con el nombre que se haya especificado.

El nombre del dispositivo puede ser:
CAS casete
CRT pantalla en modo texto
GRP pantalla en modo gráfico
LPT impresora

El nombre del programa tiene que ser una constante de cadena de la que sólo se consideran los primeros seis caracteres; si se omite, el programa se graba con un nombre que corresponde a la cadena nula. Aunque no es estrictamente necesario asignar un nombre al programa que se salva en casete, ya que tanto la escritura como la lectura en cinta se hacen de forma secuencial y entonces cada programa está, de hecho, individualizado por su posición, si es conveniente.

La instrucción SAVE graba los programas en formato ASCII; luego sobre estos ficheros se puede ejecutar la instrucción MERGE para fusionarlos con el programa presente en la memoria central.

LOAD "nombre dispositivo:[nombre fichero]" [, R]

Carga en la memoria central el fichero dado desde el dispositivo especificado. El nombre del dispositivo tiene que ser CAS; el del fichero se puede omitir, en cuyo caso se carga en la memoria el primer fichero encontrado en el casete.

Si se especifica la opción R (Run) el programa empezará a ejecutarse nada más ser cargado.

MERGE "nombre dispositivo:[nombre fichero]"

Carga desde el dispositivo dado un fichero de programa en formato ASCII y lo fusiona con el que está presente en la memoria; si el programa cargado tiene números de línea comunes con los del programa en memoria, los de éste son sustituidos por los del programa residente en memoria.

El nombre del dispositivo tiene que ser CAS; el del fichero se puede omitir, en cuyo caso carga el primer programa que encuentra en el casete.

CSAVE "nombre fichero" [, velocidad transmisión]

Graba en casete un programa en código binario; con esta instrucción el nombre del fichero no se puede omitir.

La velocidad de transmisión puede ser 1 (significa 1.200 baudios) ó 2 (representa 2.400 baudios); el valor de defecto es 1.

CLOAD ["nombre fichero"]

Carga en la memoria un fichero grabado en casete con CSAVE.

CLOAD? ["nombre fichero"]

Compara el fichero especificado del casete con el programa residente en memoria; si no son iguales aparece el mensaje "Device I/O error" o "verify error".

Se usa para comprobar que una grabación ha sido realizada correctamente después de haberla efectuado.

Si el nombre del fichero se omite, la comparación se efectúa entre el programa en memoria y el primer fichero encontrado en el casete.

BSAVE "nombre dispositivo: [nombre fichero]", dirección inicial, dirección final [, dirección principio ejecución]

Graba en formato binario un fichero, en el dispositivo indicado, con el contenido de la memoria entre dos direcciones dadas (dirección inicial y dirección final).

Un programa grabado con este comando se puede luego cargar y ejecutar directamente con el comando instrucción BLOAD y la opción R; la ejecución empezará en la "dirección principio ejecución" si está especificada; si no, empezará a partir de la dirección de principio de programa.

El nombre del dispositivo tiene que ser CAS.

Con BSAVE podemos guardar cualquier información que esté en memoria, sean datos o programas.

BLOAD "nombre dispositivo: [nombre fichero]" [, R] [, offset]

Carga en la memoria un programa grabado en casete con el comando BSAVE. Si está especificada la opción R, lo ejecuta a partir de la dirección de principio de ejecución, y si no, a partir de la primera dirección. En caso de declarar un offset éste se suma a todas las direcciones del fichero, de forma que lo podemos colocar en una parte de memoria preestablecida que nos interese.

Ficheros de datos

La primera operación necesaria para trabajar con un fichero de datos es su apertura (OPEN), que consiste en determinar el dispositivo donde se encuentra el fichero (o donde se encontrará, en el caso de que no se haya creado todavía), el nombre del fichero (opcional), la forma en la que se piensa operar sobre el fichero (lectura o escritura) y, por fin, el número del canal lógico asociado, con el cual identificaremos a partir de ese momento el fichero.

Aclaremos mejor el concepto de canal, común, por otra parte,

a cualquier ordenador. Para comunicarse con un fichero cualquiera que se encuentre en un periférico, el ordenador le reserva un área en la memoria central, de dimensiones variables según el ordenador, por la que transitan todos los datos que van o vienen del fichero; esta zona de memoria se llama *buffer de I/O*, y se conoce también con el nombre de *canal*, ya que desarrolla exactamente la función de un canal de comunicación entre la memoria central y la memoria de masa. Debido a que en un mismo programa podemos necesitar abrir simultáneamente más de un fichero, el sistema permite que usemos distintos buffers de I/O y los asocia a cada uno de los ficheros mediante un número, que es el que damos en el OPEN, para que no haya confusiones que podrían crear serios problemas.

La asociación entre un fichero y el número de canal con el que se ha abierto permanece sólo hasta el cierre del fichero, operación complementaria a la apertura que se ejecuta cuando se ha terminado de trabajar con él, y que consiste, entre otras cosas, en liberar la zona de memoria utilizada como buffer del fichero. Después del cierre del fichero, el mismo número se puede usar para referirse al canal de otro fichero, pero en ningún caso podemos tener más de un fichero abierto con el mismo número de canal.

Las demás operaciones tipicas referentes a ficheros de datos en casete son:

- escritura: consiste en la grabación de los datos del fichero en el casete;
- lectura: supone transferir los datos del fichero desde el casete a la memoria central.

Recordamos que el casete es un soporte de tipo secuencial, lo que significa que los datos se escriben uno tras otro y únicamente se pueden leer en el mismo orden de la grabación; además, para leer un dato hay que leer antes todos los que lo preceden, aunque no nos interesen. Las modalidades de lectura y escritura en casete son conceptualmente muy similares a las de escritura en impresora, que también es un dispositivo secuencial.

Como comentamos en el apartado anterior, los comandos BSAVE y BLOAD se pueden aplicar también para ficheros de datos. En el interior de un programa éstos se manejan mediante las instrucciones que veremos a continuación.

OPEN "nombre dispositivo. [nombre fichero]" FOR modo AS # núm. canal

Asocia a un fichero, localizado en el dispositivo especificado, un nombre y un número de canal y determina el modo de apertura. Los dispositivos y modos posibles son:

CAS	casete	entrada y salida
CRT	pantalla en modo texto	sólo salida
GRP	pantalla en modo gráfico	sólo salida
LPT	impresora	sólo salida

El nombre de un fichero es una cadena con un cierto número de caracteres de los que sólo se consideran significativos los seis primeros; si se omite el nombre del fichero se le asigna la cadena nula.
El modo puede ser:

OUTPUT	para escribir
INPUT	para leer
APPEND	para añadir (en escritura).

En impresora y pantalla sólo es posible el modo OUTPUT; en casete valen todos.

El número de canal debe estar comprendido entre 1 y el valor de la variable MAXFILES, ya comentada en capítulos anteriores.

Abriendo un fichero en pantalla en modo gráfico (GRP) es posible hacer aparecer caracteres alfanuméricos.

Cuando se lee un fichero desde el casete, el ordenador controla la presencia del carácter fin de fichero, grabado en el fichero cuando realizamos un CLOSE o se llega al END. Al detectarlo pone un -1 en la variable EOF (End of file, final de fichero). Si el programa no realiza la comprobación del estado de EOF y sigue intentando leer aun cuando valga -1, se producirá un error "Input past end".

El modo APPEND se utiliza para añadir datos en un fichero ya creado. Cuando hacemos un OPEN con él, se lee el fichero hasta detectar su carácter de final y los datos que se introduzcan serán grabados a partir de ese punto.

CLOSE [# número canal] [, número canal,...]

Cierra los ficheros asociados a los canales especificados, liberando así estos canales, que se pueden utilizar entonces para la comunicación con nuevos ficheros.

Si no se especifica ningún número de canal, se cierran todos los ficheros.

PRINT # número canal; [, expresión separador expresión...]

Escribe uno o más datos en el fichero abierto con el número de canal especificado.

El funcionamiento es como el de PRINT por lo que se refiere

a impresora y pantalla, pero en la escritura sobre cinta hay alguna diferencia.

Si queremos escribir distintos datos alfanuméricos en la misma línea, de forma que luego se puedan volver a leer individualmente, deberemos separarlos con comas, que hay que declarar explícitamente en la instrucción PRINT; por ejemplo, la instrucción
PRINT #2, A$",";B$
escribe dos valores distintos A$ y B$ mientras que

PRINT #2, A$ B$

escribe un único valor de cadena suma del contenido de A$ y B$ con lo cual al realizar la lectura no será posible leer de forma separada A$ y B$

Por lo que se refiere a las variables numéricas, en cambio, el ordenador inserta automáticamente en la cinta una coma entre una y otra.

PRINT # número canal USING "formato"; expresión [, expresión...]

Funciona como PRINT USING, pero escribiendo los datos en el fichero abierto con el número de canal especificado.

INPUT # número canal, variable [, variable,...]

Lee las variables del fichero abierto con el número de canal especificado.

LINE INPUT # número canal, variable de cadena

Mete en la variable de cadena todos los caracteres que lee en el fichero, desde el punto en el que se encuentra hasta el primer código de RETURN CHR$(13).

CAPITULO VII

TRATAMIENTO DE LAS INTERRUPCIONES



Con el término interrupción se entiende, en general, el mecanismo mediante el cual se puede suspender la ejecución del programa en curso como consecuencia de un evento externo al mismo programa y que requiere un tratamiento particular e inmediato. Mediante el tratamiento de la interrupción permitimos que el ordenador "reaccione" de forma adecuada y a tiempo al suceso.

En el caso del BASIC MSX, las circunstancias que pueden causar una interrupción son:

- verificarse de un error;
- pulsación de una tecla de función,
- pulsación de la barra espaciadora o del botón de uno de los dos joysticks;
- pulsación simultánea de las teclas CTRL y STOP,
- superposición (choque) de dos sprites;
- final de un período predeterminado.

La mecánica es parecida en todos los casos, así que la explicaremos solamente para uno, concretamente para la pulsación simultánea de CTRL y STOP.

Situando una instrucción del tipo

ON STOP GO SUB...

en el programa declaramos el número de línea a donde debe acudir el programa cuando se produzca la interrupción debida a

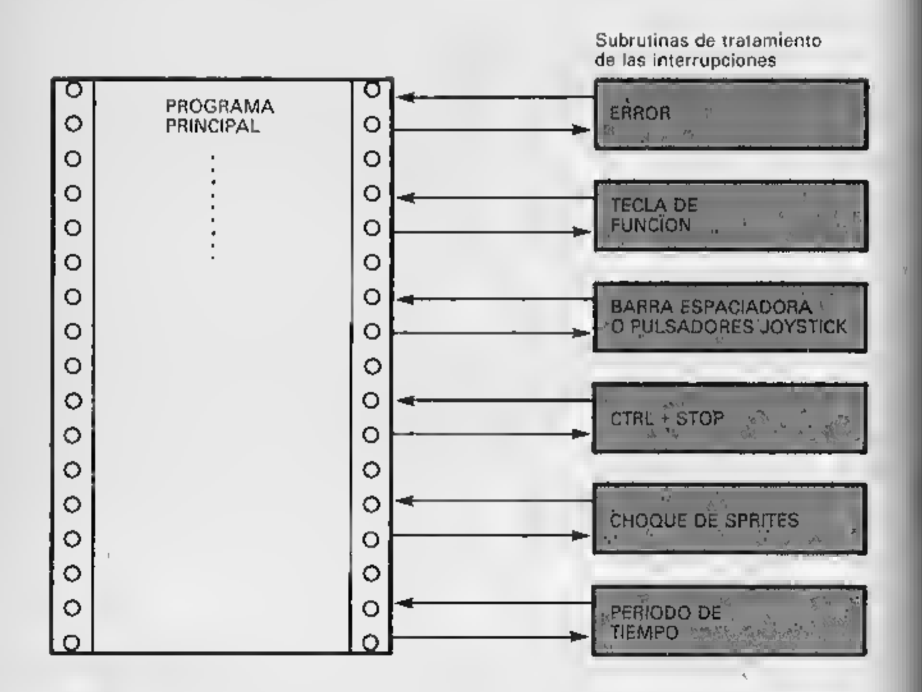


Fig. 1 - Diversas causas que pueden provocar la cesión del control a una subrutina de tratamiento de interrupciones. Para que se produzca este hecho la interrupción concreta que se quiera atender debe estar habilitada.

CTRL+STOP. En ese punto deberá comenzar la subrutina de tratamiento de esta interrupción.

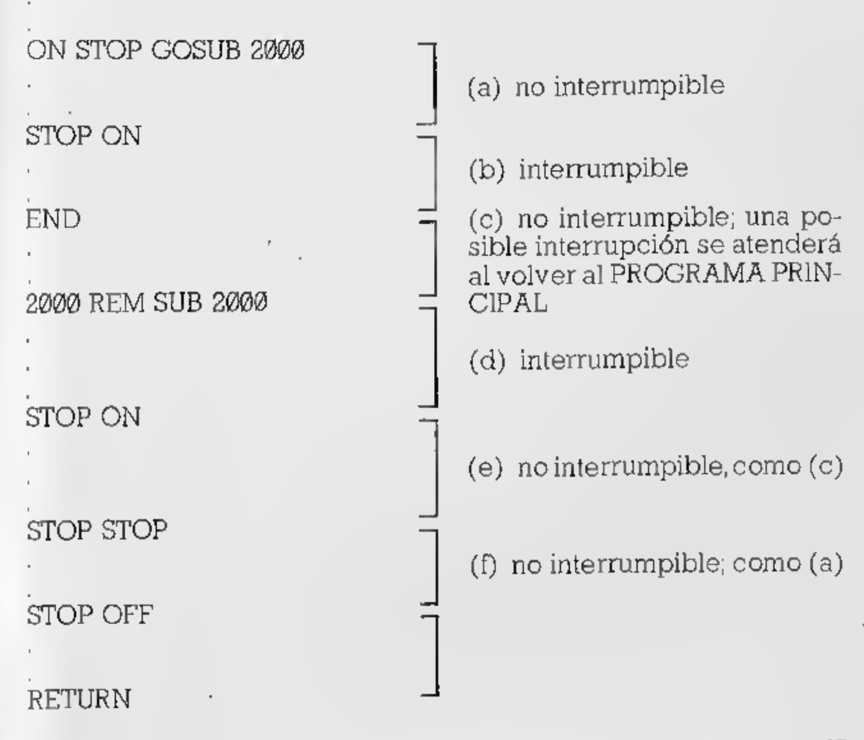
A partir de este momento la pulsación de estas dos teclas puede provocar una interrupción del programa y el salto a la subrutina que se encuentra en la línea especificada. Al final de la subrutina el control vuelve al programa principal, a la instrucción que había sido interrumpida o a la que se especifique en el RETURN.

Ahora bien, para que la interrupción tenga realmente lugar hay que dar antes una instrucción de STOP ON: a partir de ese momento y hasta que aparezca la instrucción contraria (STOP OFF), la pulsación de CTRL+STOP provocará, efectivamente, la interrupción del programa.

Generalmente, una subrutina de servicio no se puede interrumpir, ya que las interrupciones son deshabilitadas internamente hasta la vuelta al programa principal. Una señal de interrupción

que se verificase antes se memorizaría y sería atendida cuando se produjera el retorno al programa principal. Ahora bien, si aparece una instrucción STOP ON en la subrutina de servicio, la inhabilitación de la interrupción se olvida y se podrá interrumpir también inmediatamente la subrutina. En el caso opuesto, es decir, si incluimos una instrucción STOP OFF, las interrupciones se ignorarán totalmente y no se atenderán ni siquiera al final de la subrutina cuando regresemos al programa principal.

Si en una subrutina de servicio hemos dado un STOP ON haciéndola de esta forma interrumpible, podemos anular esta orden en cualquier otro punto de la rutina con STOP STOP; de esta forma la subrutina se puede dividir en secciones interrumpibles y no. La diferencia entre STOP STOP y STOP OFF es que la primera sólo anula el STOP ON anterior, llevando a la subrutina de servicio al mismo estado de deshabilitación interna inicial, en tanto el STOP OFF provocaría las variacionesn antes señaladas.



```
REM PROGRAMA PRINCIPAL
.
.
ON STOP GOSUB 2000        ] (a) no interrumpible
.
.
STOP ON                   ] (b) interrumpible
.
.
END                       ] (c) no interrumpible; una posible interrupción se atenderá al volver al PROGRAMA PRINCIPAL
.
2000 REM SUB 2000         ] (d) interrumpible
.
.
.
STOP ON                   ] (e) no interrumpible, como (c)
.
.
.
STOP STOP                 ] (f) no interrumpible; como (a)
.
.
STOP OFF
.
.
RETURN
```

Las interrupciones debidas a las demás circunstancias citadas funcionan de la misma forma, excepto el caso de los errores, que describiremos detalladamente con la instrucción ON ERROR GOTO.

Una vez explicado el proceso veamos ahora las distintas instrucciones que permiten llevarlo a cabo.

ON KEY GOSUB número de línea [, número de línea,...]

Declara los números de línea donde empiezan las subrutinas que atienden las interrupciones debidas a la pulsación de una tecla de función concreta.

Puede haber tantos números como teclas de función queramos atender. La primera subrutina se ejecuta cuando pulsamos la tecla F1, la segunda cuando pulsamos F2, y así sucesivamente.

KEY (n) ON
KEY (n) OFF
KEY (n) STOP

Respectivamente habilitan, deshabilitan definitivamente o suspenden hasta el final de la subrutina de servicio las interrupciones debidas a la presión de la tecla de función Fn.

ON STRING GOSUB número de línea [, número de línea,...]

Declara los números de línea donde empiezan las subrutinas de servicio de las interrupciones debidas, respectivamente, a la pulsación de:

- la barra espaciadora,
- el botón 1 del joystick 1,
- el botón 1 del joystick 2,
- el botón 2 del joystick 1,
- el botón 2 del joystick 2.

STRING (n) ON
STRING (n) OFF
STRING (n) STOP

Respectivamente habilitan, deshabilitan definitivamente o suspenden hasta el final de la subrutina de servicio la interrupción debida a la tecla n, donde:

n=1: barra espaciadora
n=2: botón 1 joystick 1

n=3: botón 1 joystick 2
n=4: botón 2 joystick 1
n=5: botón 2 joystick 2

ON STOP GOSUB número de línea

Declara el número de línea donde comienza la subrutina de servicio de la interrupción debida a la pulsación de CTRL y STOP.

STOP ON
STOP OFF
STOP STOP

Respectivamente, deshabilitan definitivamente o suspenden hasta el final de la rutina de servicio las interrupciones debidas a la pulsación de las teclas CTRL y STOP.

ON SPRITE GOSUB número de línea

Declara el número de línea de la subrutina de servicio de la interrupción provocada por la superposición (choque) de dos sprites.

SPRITE ON
SPRITE OFF
SPRITE STOP

Respectivamente habilitan, deshabilitan o suspenden hasta el final de la subrutina de servicio las interrupciones debidas a la superposición de dos sprites.

ON INTERVAL=intervalo GOSUB número de línea

Declara el número de línea de la subrutina de servicio a la que el programa tiene que saltar al cabo de un cierto período de tiempo definido por el valor de "intervalo", que se expresa en 1/5Ø de segundo (un intervalo igual a 5Ø representa un tiempo de 1 segundo). Mientras que no la desactivemos, en cada "intervalo" se producirá esta interrupción.

INTERVAL ON
INTERVAL OFF
INTERVAL STOP

Respectivamente habilitan, deshabilitan definitivamente o suspenden hasta el final de la subrutina de servicio las interrup-

ciones debidas al cumplimiento del periodo de tiempo establecido.

ON ERROR GOTO número de línea

Declara el número de línea al que debe saltar el programa si se verifica un error. Además se encarga también de habilitar las interrupciones debidas a este evento.

Normalmente, al verificarse un error durante la ejecución de un programa, se bloquea la ejecución de éste y se produce la visualización del oportuno mensaje. Con esta instrucción, en cambio, cuando se verifica un error el programa salta a la rutina especificada que, naturalmente, tiene que ser escrita por el usuario. En esta rutina, por medio de un control sobre el código de error (ERR) y sobre el número de línea en el que se ha detectado (ERL), se puede identificar el error y tomar las medidas oportunas para evitar la interrupción de la ejecución.

RESUME Ø
RESUME NEXT
RESUME número de línea

Es el equivalente del RETURN para la subrutina de servicio del error. Devuelve el control, respectivamente, a la línea en la que ha ocurrido el error, a la siguiente o a la línea especificada.

ON ERROR GOTO Ø

Deshabilita las interrupciones debidas a errores. Por tanto, desde ese momento el sistema se encargará, en caso de error, de generar el habitual mensaje e interrumpir el programa.



SUBRUTINAS Y PROGRAMAS DE APLICACION



n este capítulo les presentaremos algunos programas y subrutinas interesantes que usted podrá estudiar para comprender mejor el funcionamiento de las instrucciones del BASIC MSX que hemos descrito en los capítulos anteriores y que, además, podrá también utilizar tal y como están. Son subrutinas sencillas que realizan trabajos útiles en muchos contextos distintos, desde el programa de gráficos al de contabilidad, y cuyo conocimiento le puede ahorrar un tiempo considerable a la hora de llevar a cabo el diseño de sus programas.

En el transcurso del capítulo analizaremos también algunos problemas de carácter general relativos al "buen estilo" de la programación, con algunos consejos útiles sobre cómo mejorar la eficacia y la legibilidad de los programas que escriba.

Esperando la pulsación de una tecla

```
520 REM espera la pulsación
530 REM    de una tecla
540 REM ----------------------
550 A$="< PULSA UNA TECLA PARA CONTINUAR >"
560 GOSUB 1390
570 IF INKEY$="" THEN 570
```

Las instrucciones 55Ø-57Ø realizan una función muy común. En la ejecución de este segmento de programa el ordenador se pone en estado de espera; en la pantalla se visualiza el mensaje que

contiene A$ (a través de la subrutina 1360 que veremos después) y la ejecución del programa se para hasta que pulsemos una tecla cualquiera. Las líneas 550 y 560 visualizan la frase que informa que hay que pulsar una tecla; la subrutina que empieza en la línea 1390 se describe en un próximo apartado, pero adelantamos que sólo sirve para centrar y visualizar en la pantalla el contenido de A$

De hecho es la línea 570 la que hace todo el trabajo. La función INKEY$ controla si se ha pulsado una tecla y toma el valor de la cadena nula (indicado por una pareja de comillas que no contienen nada) si no ha ocurrido esto. Se hace un test sobre el valor que devuelve INKEY$ con la instrucción IF..THEN. Si la condición INKEY$="" es verdadera, o sea, si no se ha pulsado ninguna tecla, entonces (THEN) se vuelve a la línea 570 esperando que ocurra; de otra forma, pasa a la instrucción siguiente.

Para comprender mejor el sentido de esta instrucción le aconsejamos que ejecute el programa después de haber dado el comando TRON. Con dicho comando se visualizan los números de línea de las instrucciones, entre paréntesis cuadrados, según se van ejecutando. Podrá así darse cuenta de los continuos [570] que pasan por la pantalla; efectivamente, el ordenador seguirá ejecutando la línea 570 hasta que no pulsemos una tecla.

Programas amigables y hostiles

Un buen programador debe ser capaz de hacer cómoda la tarea a quien tenga que usar el ordenador, y para ello debe cuidar de forma muy especial las instrucciones de comunicación con el usuario. Saber facilitar el uso de un programa significa que el propio programa (o el manual que lo acompaña, si es muy complejo) sean lo suficientemente claros a la hora de explicar al usuario cómo debe hacer uso de él y de qué forma manejarlo. Un programa diseñado con estos criterios se dice "user friendly", o sea, amigable para el usuario porque trata de ayudarle.

Para aclarar la diferencia existente entre un programa amigable y uno hostil, le proponemos dos ejemplos extremos que hacen exactamente lo mismo: aceptan un número comprendido entre 1 y 5. Piénselos como parte de un programa mucho más grande, en el que el número elegido tiene considerable importancia (es el código secreto de una caja fuerte que contiene documentos fundamentales para la seguridad nacional...).

Pruebe los dos programas en su ordenador y teclee números casuales para ver cómo funcionan.

1. Programa hostil

```
10 INPUT "TECLEA UN NUMERO ";A
20 IF (A<1) OR (A>5) THEN BEEP:BEEP:BEEP:PRINT
"EQUIVOCADO!":GOTO 10
```

¿Muy grosero, no le parece?
El mismo programa en versión amiga:

2. Programa amigable

```
10 INPUT "TECLEA UN NUMERO ENTRE 1 Y 5 ";A
20 IF (A<1) OR (A>5) THEN BEEP:PRINT "Lo siento,
te has equivocado. Inténtalo otra vez":GOTO 10
```

Este programa, por lo menos, explica en la línea 10 cuáles son los números que podemos pulsar; este detalle, aparentemente insignificante, puede ahorrar mucho tiempo y muchas frustraciones. Por otra parte, la función de un sistema para el control de los errores es señalar eventuales equivocaciones, y no pretender dar una lección al que está usando el ordenador.

Centrado de las cadenas

```
1360 REM  subrutina de centrado
1370 REM      de las cadenas
1380 REM ----------------------
1390 A=LEN (A$)
1400 B=INT((37-A)/2)
1410 PRINT TAB(B);A$
1420 RETURN
```

Esta subrutina tiene como función centrar el mensaje que contiene la variable de cadena A$ ¿Cómo?

La línea 1390 calcula el número de caracteres de A$ mediante la instrucción LEN y los asocia a B. Calcula entonces el número de espacios en blanco que quedarían en su pantalla (supone que es de 37 caracteres; si no fuera así, bastaría cambiar este número) al escribir el mensaje y pone delante la mitad de dicho número, truncando antes por si no fuera par.

La función INT devuelve el número entero menor o igual que el considerado. Así, INT (23,17) = 23, INT (23,95) = 23. La función TAB sitúa el cursor en el punto correspondiente, y a partir de él escribe el mensaje. Finalmente, el RETURN devuelve el control a la línea siguiente a aquella desde donde se realizó el GOSUB.

Creación de un marco

```
1470 REM  subrutina para el trazado
1480 REM       de un marco
1490 REM  ----------------------------------
1500 FOR I=1 TO 18
1510   PRINT"*  ";
1520   FOR J=1 TO 30
1530   REM no hace nada
1540   NEXT J
1550 NEXT I
1560 PRINT:PRINT
1580 RETURN
```

Esta subrutina permite trazar en la pantalla un marco con el carácter que se especifica entre comillas en el PRINT de la línea 151Ø. Debido a que la anchura por defecto de la pantalla es de 37 columnas, el carácter, conjuntamente con un espacio, se repite 18 veces por medio de un ciclo FOR...NEXT que empieza en la línea 150Ø y termina en la línea 155Ø. En el interior de este ciclo está presente otro, cuya tarea es simplemente la de introducir cierto retardo en la visualización de los caracteres para hacer más atractiva la presentación. Los dos PRINT de la linea 156Ø tienen como único fin el de separar la linea de asteriscos (o de cualquier otro carácter que haya elegido) del texto que sigue; la subrutina termina con RETURN, que devuelve el control al programa principal. Llamando esta subrutina repetidamente puede resaltar parte del texto o los comentarios en el interior de sus programas.

Salida del programa

```
1630 REM   subrutina de salida
1640 REM       del programa
1650 REM  --------------
1660 CLS
1670 LOCATE 1,10
1680 A$="ADIOS !!!"
1690 GOSUB 1390
1700 PRINT
1720 END
```

Siempre es bueno introducir una subrutina de salida de un programa para señalar su final normal. La que hemos introducido aquí es del tipo más simple que se pueda imaginar: su única función es la de escribir "ADIOS !!!" en el centro de la pantalla. En ocasiones puede convenir que la subrutina de salida dé alguna información de cómo apagar el ordenador o cómo volver a ejecutar el programa. En algunos programas sofisticados incluso borra todas las instrucciones de la memoria de forma que no sea posible ver el listado del programa.

Nuestra subrutina borra la pantalla (lína 166Ø), posiciona el cursor al principio de la décima línea (instrucción 167Ø) y asigna a la variable de cadena A$ los caracteres que se quieren visualizar. La llamada a la rutina 139Ø (apartado "Centrado de las cadenas") permite escribir el valor contenido en A$ en una posición centrada con relación a los márgenes horizontales de la pantalla. Obviamente, la subrutina de salida del programa no termina con RETURN sino con END, que pone fin a la ejecución.

Control de los datos de entrada

```
650 REM solicita los operandos y la
660 REM     operación a ejecutar
670 REM  --------------------------------------
680 CLS
690 LOCATE 3,1
700 INPUT "PRIMER OPERANDO  --- ";O1$
710 O1=VAL(O1$)
720 IF O1=0 GOTO 690
730 LOCATE 3,4
740 INPUT "SEGUNDO OPERANDO --- ";O2$
750 O2=VAL(O2$)
760 IF O2=0 GOTO 730
770 LOCATE 3,5
780 LINE INPUT "OPERACION   --- ";OP$
```

Suponga que queremos introducir en un programa dos valores numéricos y el símbolo de una operación aritmética (+, −, ×, /) que se va a ejecutar con ellos.

Un buen programa tiene que ser "fool proof", como dicen los americanos (nosotros diríamos a prueba de tontos), en el sentido que ni siquiera pulsando datos equivocados en el teclado se tienen que verificar errores irrecuperables. Normalmente, quien usa un programa lo debería hacer de forma correcta, pero para prevenir errores debidos a inexperiencia, distracción o "mala idea" es necesario introducir controles sobre los datos introducidos, después analizaremos los que usa este programa. Las líneas 68Ø-78Ø se ocupan de la introducción de los datos necesarios para el cálculo: primero los dos operandos y luego la operación que se quiere realizar. Mientras que los operandos se introducen con

operaciones normales de INPUT, el tipo de operación se toma mediante un LINE INPUT que acepta todos los caracteres alfanuméricos que aparezcan hasta el RETURN, con un máximo de 255. En ambos casos las instrucciones de entradas (input) están acompañadas por un "prompt", o sea, por un mensaje de petición. Es conveniente hacer uso de estos avisos, pues hacen más fácil el uso del programa y reducen las posibilidades de error.

Para prevenir posibles equivocaciones al teclear los operandos el ordenador los toma como cadenas de caracteres para después convertirlas en el correspondiente valor numérico con la instrucción VAL. Así, si el primer carácter no es numérico, el valor que se obtiene es Ø, y en ese caso se vuelve a pedir el valor del operando en lugar de bloquearse la ejecución y dar un mensaje de error, que sería lo que ocurriría si no hubiésemos puesto el control.

Esta técnica no impide del todo que surjan los errores, pues, por ejemplo, tecleando 12AA34 el ordenador lo interpretará como 12. Intente, en base a lo visto, inventar una subrutina que controle de forma completa los datos en entrada (puede ser un ejercicio útil y la podrá usar en todos los casos en los que se necesiten datos numéricos).

Veamos ahora cuál sería un posible control sobre la operación elegida.

```
820 REM controla la validez
830 REM    de la operación
840 REM ----------------------
850 IF (OP$="+") THEN RI=O1+O2:GOTO 990
860 IF (OP$="-") THEN RI=O1-O2:GOTO 990
870 IF (OP$="*") THEN RI=O1*O2:GOTO 990
880 IF (OP$="/") THEN RI=O1/O2:GOTO 990
890 REM
900 REM  no es una de las
910 REM cuatro operaciones
920 REM ------------------
930 LOCATE 1,21
940 PRINT"Sólo acepto cuatro operaciones
prefijadas: + - * /"
950 GOTO 770
```

Observe que la función de las instrucciones 85Ø-88Ø es la de controlar si las operaciones introducidas entran dentro de las que el ordenador está capacitado para hacer y, en ese caso, calcular al mismo tiempo el resultado.

La técnica adoptada es la de los controles en cascada, es decir, una serie de IF... THEN que examinan todas las posibilidades.

Si todos los controles dan resultado negativo se imprime un mensaje de error y el programa vuelve a pedir el tipo de operación que se quiere ejecutar.

Si el símbolo introducido es, en cambio, uno de los cuatro admitidos, la ejecución continuaría desde otra línea (en el ejemplo, la 99Ø).

Otro tipo de control, que se podría establecer, por ejemplo, ante situaciones como "1 = salida a pantalla, 2 = salida por impresora", podría ser:

```
640 REM   comprueba la validez
650 REM    de la contestación
660 REM ----------------------
670 REM
680 PRINT"Número de tu opción = ";
690 X$=INPUT$(1)
700 IF (X$<>"1") AND (X$<>"2") THEN CLS:
GOTO 680
```

La línea 7ØØ comprueba si el dato de entrada está dentro del rango admitido. La selección se hace a través del IF..THEN, que en este caso valora una expresión lógica entera; las condiciones de no validez, X$<>"1" y X$<>"2", están unidas por el operador lógico AND. Si la expresión lógica resulta ser falsa (la variable es válida), la ejecución continúa a partir de la instrucción siguiente al IFTHEN; de otra forma se ejecutan las instrucciones que siguen al THEN (en este caso se repetiría la petición). Los resultados devueltos por el operador AND se obtienen en función de los valores de las dos relaciones según la siguiente tabla de la verdad:

X$<>"1" X$<>"2"	verdadero verdadero	verdadero falso	falso verdadero	falso falso
X$<>"1" AND X$<>"2"	verdadero	falso	falso	falso

Resumiendo: si X$ no es ni 1 ni 2, el programa no acepta la contestación del operador, borra la pantalla y vuelve a la línea 68Ø para proponerle que defina de nuevo su elección. Observe que X$ es una variable de cadena (la instrucción INPUT$ proporciona sólo variables de cadena) y que entonces, en la comparación, es necesario delimitar los números 1 y 2 entre comillas de forma que el ordenador los trate como constantes alfanuméricas y no numéricas.

Legibilidad y REM

Los listados de las subrutinas que encontrará en este libro se comentan abundantemente con instrucciones REM, que se pueden eliminar sin perjudicar el funcionamiento del programa. Efectivamente, REM es una instrucción que el ordenador no ejecuta; sólo sirve al autor del programa para intercalar notas (REM viene de REMarks) que aclaren la estructura del programa y le faciliten las correcciones o ampliaciones.

Para evitar problemas y mensajes de error a la hora de usar una versión donde hubiéramos quitado todos los REM, las direcciones de llamada a las subrutinas y las usadas en los GOTO son las correspondientes a la primera línea ejecutable (no a los REM) de la subrutina o segmento de programa.

Es una buena norma de programación comentar con claridad los programas, tanto para su posterior depuración (eliminación de errores) como para poderlos modificar al cabo del tiempo sin tener que perder horas y horas reconstruyendo el funcionamiento del programa. Esto ocurre, aunque usted crea que no, pues los programas en BASIC tienen la antipática característica de volverse ilegibles al poco tiempo, inclusive para el programador que los ha escrito (si usted piensa que con ese programa que acaba de escribir no le pasará, deje pasar algún tiempo y ya nos contará). La instrucción REM mejora de forma sensible la claridad y la legibilidad del listado, permitiendo modificaciones aun después de mucho tiempo.

También es útil colocar al principio del programa alguna información auxiliar, como el nombre del programa, quién y cuándo lo ha escrito y si está protegido por un copyright.

La lógica de los ordenadores

Los ordenadores trabajan, internamente con números binarios, que sólo pueden tener dos valores: 1 y Ø. La lógica en la que se basan a la hora de tomar decisiones es también de dos valores: sólo existen el verdadero y el falso; una proposición sólo puede ser verdadera o falsa, no existen otras posibilidades. Una lógica de este tipo se dice Booleana, porque fue el matemático George Boole el que la formalizó en el siglo pasado. Los operadores lógicos (AND, OR, NOT, XOR) permiten efectuar operaciones con los valores Booleanos (verdadero y falso) de la misma forma que hacemos con los operadores aritméticos y los números decimales, a tal punto que se puede construir un álgebra basada en dichos operadores.

Señalización de una contestación acertada o equívocada

```
1600 REM  subrutina para el caso de
1610 REM     contestación exacta
1620 REM ---------------------------------------
1630 REM
1640 BEEP
1650 PLAY "V12AGEF"
1660 G=G + 1
1670 PRINT
1680 PRINT
1690 PRINT"Felicidades, has contestado de forma"
1700 PRINT
1710 PRINT"exacta!"
1720 LOCATE 1,22
1730 A$=PULSA UNA TECLA PARA CONTINUAR":GOSUB 1240
1740 IF INKEY$="" THEN 1740
1750 RETURN
1760 REM
1770 REM =======================================
1780 REM  subrutina para el caso de
1790 REM   contestación equivocada
1800 REM ---------------------------------------
1810 REM
1820 REM
1830 BEEP
1840 PLAY "V12AAABBB"
1850 S=S +1
1860 PRINT
1870 PRINT
1880 PRINT"Lo siento, te has equivocado"
1890 RETURN
```

En muchos programas es útil señalar, tanto visible como acústicamente, si la contestación proporcionada es aceptable para el ordenador. Estas dos subrutinas, muy simples, cumplen con el requisito. Si la contestación es correcta se debe ejecutar la subrutina 164Ø; si no, la 183Ø. La estructura de ambas subrutinas es similar: las dos visualizan un mensaje (distinto según el caso) y hacen oír una nota para llamar la atención del usuario. Para que suene el pitido en las dos rutinas hemos utilizado la instrucción PLAY, que ya se explicó. Aquí, en concreto, hemos variado el volumen del sonido con el comando V12, y, por ejemplo, en la línea 165Ø se tocan las notas La y Si (la notación utilizada por el PLAY es la americana, por lo que La=A y Si=B). La instrucción BEEP que precede al PLAY no es indispensable, pero lleva todos los valores

de los registros de sonido a su valor de defecto, sin preocuparse por las modificaciones que se hayan realizado anteriormente.

Siempre que se acceda a una de estas subrutinas se incrementará una variable (en el primer caso es G y en el segundo S) para llevar la cuenta del número de contestaciones correctas y equivocadas.

La subrutina 1240 que se llama en 1730 es la ya conocida para centrar las cadenas.

Las interrupciones

```
630 REM activa las interrupciones de las
640 REM  teclas de función F1, F2, F3
650 REM ------------------------------------
660 REM
670 ON KEY GOSUB 2030,2260,2460
680 KEY(1) ON
690 KEY(2) ON
700 KEY(3) ON
```

Las interrupciones son una de las características más destacadas del estándar MSX, ya que dan al programador grandes posibilidades. Una aplicación típica es la ejecución de una rutina, internamente a un programa principal, al verificarse un evento (la pulsación de una tecla, el choque de dos sprites, etc.) que tiene un carácter ocasional.

En el corto y simple segmento de programa reproducido anteriormente se representa la activación de las interrupciones para las teclas de función F1, F2 y F3, cuya pulsación hará que el control se pase a las tres subrutinas indicadas en la línea 670, que tomarán las acciones oportunas que el programa específico determine.

El nombre de las variables

Las variables en BASIC no están sujetas a reglas muy restrictivas: no es necesario declarar al principio del programa ni cuántas ni cuáles ni de qué tipo se van a usar. Si en un determinado punto de la ejecución aparece una nueva variable (aunque sea debida, por ejemplo, a un error al teclearla), el programa no se asusta y la emplea tranquilamente, asignándola el valor inicial 0 si es una variable numérica, o la cadena nula si es alfanumérica.

Por lo que se refiere al nombre, teóricamente se pueden usar cualesquiera nombres que queramos, pero es preferible que tengan alguna relación con el significado de la variable. Por ejemplo, si tenemos dos números, A y B, y queremos calcular el valor medio, es aconsejable usar la variable MEDIA=(A+B)/2 antes que la expresión "anónima" C=(A+B)/2. Sin embargo, hay que tener cuidado, ya que el BASIC MSX, como la mayoría de los dialectos BASIC utilizados en los ordenadores personales y domésticos, reconoce sólo las primeras dos letras del nombre de la variable. Esto supondría que las variables XMINIMO y XMAXIMO serían, en realidad, la misma variable XM para el ordenador, y habría que acudir, por ejemplo, a XINF y XSUP.

La otra limitación está representada por la imposibilidad de usar como nombres de variables, o como parte de ellos, las palabras reservadas del BASIC. Sería muy cómodo llamar MIN y MAX a las variables destinadas a contener los valores extremos, pero, siendo MAXFILES un comando, tenemos que recurrir a XINF y XSUP.

Formateado de la pantalla.

Formatear es una palabra de la jerga informática, tomada del inglés formatting, a la que se atribuyen distintos significados. Uno de ellos hace referencia a la presentación, de forma clara y ordenada, de mensajes e instrucciones en la pantalla, es decir, a la organización de los formatos de salida de la información. La importancia del formateado es evidente: se tarda menos en seguir instrucciones precisas que no en volverse loco para entender qué puede significar una frase oscura o un punto interrogante puesto aleatoriamente en la pantalla.

Para determinar el formato de la información, el BASIC MSX dispone de una serie de instrucciones muy poderosas:

- Para posicionar el cursor:
  TAB (x)
  SOC (x)
  LOCATE x, y
  ; (hace que el cursor se quede en la misma línea del último PRINT)
  , (hace que el cursor pase a la zona siguiente; las zonas son de 14 caracteres)

- Para escribir:
  PRINT
  PRINT USING
  LPRINT (sólo si está conectada la impresora)
  LPRINT USING (sólo si está conectada la impresora)

La pantalla

La pantalla de los ordenadores MSX está gestionada por un procesador de video muy sofisticado que controla todas las operaciones referentes al dibujo o a la escritura en la pantalla de televisión o en el monitor. Hasta la simple visualización de una letra implica toda una serie de operaciones complicadas y muy rápidas que sólo un dispositivo muy desarrollado puede llevar a cabo. Existen cuatro modalidades de utilización de la pantalla:

SCREEN Ø

Selecciona la modalidad de texto con 24 líneas × 37 columnas; con el comando WIDTH 40 puede aumentar el número de caracteres visualizados hasta 40. Cada carácter está constituido por una matriz de 6×8 puntos, por lo cual algunos caracteres gráficos, que necesitan 8×8 puntos, no se visualizan correctamente.

SCREEN 1

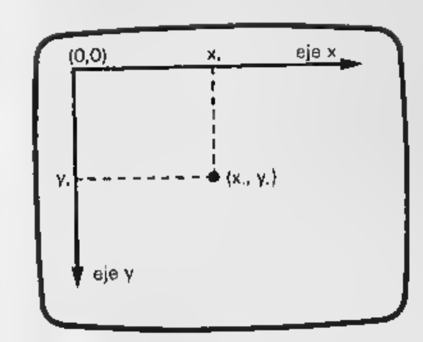
Selecciona la modalidad de texto con 24 líneas × 29 columnas; con el comando WIDTH 32 puede aumentar el número de caracteres hasta 32. La matriz del carácter es de 8×8 puntos, lo que permite representar correctamente todos los caracteres disponibles en el teclado.

SCREEN 2

Selecciona la modalidad gráfica de alta resolución (256×192 puntos), con algunas limitaciones en el color: no podemos elegir de forma independiente el color de una serie de 8 puntos, sino que estamos obligados a elegir un máximo de dos colores; si especificamos un tercero, todos los 8 puntos asumirán el nuevo color.

Para la pantalla gráfica hay que subrayar algunos aspectos; el origen de coordenadas se encuentra en la esquina superior izquierda (Fig. 1). Si vamos hacia abajo en la pantalla, el valor de las "y" aumenta, en lugar de disminuir, como pasa normalmente. Atención entonces: para dibujar un punto de coordenadas cartesianas "normales" (x', y') habrá que especificar en su MSX (Fig. 1b) las coordenadas (x', 192-y').

La existencia de más pixels en la dirección horizontal que en la vertical implica unas notables deformaciones visuales de las figuras: una línea recta a 45 grados parece tener una inclinación distinta, un círculo parece una elipse, etc. Para solucionar este inconveniente es suficiente multiplicar el valor de y por 256/192, que es la relación entre los puntos en las dos direcciones.



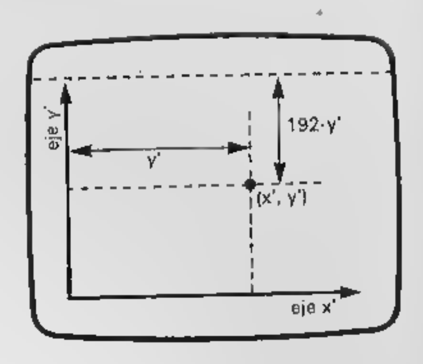


Fig. 1 - a) Ejes usados en la pantalla por los ordenadores MSX. b) Conversión de las coordenadas cartesianas habituales de un punto a las equivalentes para el MSX.

SCREEN 3

Selecciona el modo multicolor, que usa una modalidad gráfica de resolución menor (64×48 bloques de 4×4 puntos cada uno), y en la que podemos elegir el color que más nos guste en una gama de 16 disponibles.

Números aleatorios

Los números generados por el ordenador con la instrucción RND no son verdaderos números aleatorios, pues, aunque tras mucho tiempo, acaban repitiéndose. La razón por la que parecen aleatorios es, precisamente, que la cantidad de números entre los que la instrucción RND elige es tan grande que hace muy improbable que en dos extracciones consecutivas aparezca el mismo número. Para que la elección sea aún más aleatoria puede realizar cada llamada a RND con una instrucción del tipo:

```
R=RND(-TIME)
```

donde la variable R es ficticia; lo que importa es la RND(-TIME), que genera una nueva secuencia de números partiendo de la variable del sistema TIME, que contiene el tiempo transcurrido desde el encendido del ordenador, expresado en 1/50 de segundo.

```
730 REM definición de los sprite
740 REM -----------------------------
750 REM
760 CLS:COLOR 4,15,0
770 SCREEN 2,0
780 FOR I=1 TO 5
790   A$=""
800   FOR J=1 TO 8
810   A$=A$+CHR$(A)
830   NEXT J
840   SPRITE$(I)=A$
850 NEXT I
2830 REM """""""""""""""""""""""""""""""
2840 REM "                              "
2850 REM "  DATA PARA LOS SPRITE        "
2860 REM "                              "
2870 REM """""""""""""""""""""""""""""""
2880 REM
2890 REM *** PROYECTIL ***
2900 REM
2910 DATA 0,0,24,60
2920 DATA 60,24,0,0
2930 REM
2940 REM *** DIANA ***
2950 REM
2960 DATA 0,0,0,1,3,7,127,255
2970 DATA 255,127,7,3,1,0,0,0
2980 DATA 0,64,192,192,193,195,255,255
```

El primer paso es definir los sprites que se quieren utilizar. Les aconsejamos dibujarlos antes en una hoja cuadriculada, utilizando bloques de 8×8 o de 16×16 cuadraditos, según las dimensiones deseadas, y luego convertir el dibujo en una serie de números decimales. La técnica adoptada en el programa es la de escribir los valores decimales que definen un sprite en una instrucción DATA y luego introducir una subrutina (como la de las líneas 760-830) que crea el sprite.

La definición del sprite siempre tiene que estar precedida por el paso a pantalla gráfica con la instrucción SCREEN 2 seguida por una coma y por un número que especifica las características dimensionales del sprite, tal y como explicamos en el capítulo dedicado a los gráficos.

Partiendo del dibujo sobre papel cuadriculado hay que escribir un número binario que describa el sprite. Este primer paso es muy sencillo: cada cuadradito que hallamos pintado de negro vale 1 y cada uno sin pintar vale Ø. De esta forma se obtiene una serie de números binarios que es necesario convertir a decimal o hexadecimal.

Para pasar un número binario a hexadecimal agrúpelo de 4 en 4 bits empezando por el situado más a la derecha:

1010011 ⟶ 101.0011

sustituya ahora cada uno de estos bloques por su correspondiente equivalente hexadecimal (vea, por ejemplo, la tabla adjunta).

. . 101.0011 ⟶ 53 en hexadecimal

Para pasar un número binario a decimal. Asignando el valor Ø a la posición que ocupa el bit más a la derecha y los sucesivos (1, 2,...) a los situados a su izquierda, calcule la suma que resulta de aplicar la fórmula:

(valor bit de la posición i)$\times 2^i$

a todos y cada uno de los bits empleados.

1010011 ⟶ $1\times2^0+1\times2^1+Ø\times2^2+Ø\times2^3+1\times2^4+Ø\times2^5+1\times2^6=83$ en decimal.

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	O
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11
18	10010	22	12
19	10011	23	13

Los números decimales metidos en los DATA se leen en bloques de 8 mediante un ciclo de FOR...NEST que, de esta forma, construye una variable de cadena A$ que luego pasa a la instrucción SPRITE$ que define el sprite. El procedimiento se puede repetir con un segundo ciclo FOR...NEXT para todos los sprite deseados (con tal de que no superen el límite máximo de 255 si el parámetro usado en SCREEN es Ø ó 1, o de 63 si el parámetro vale 2 ó 3).

Ya que las dimensiones del sprite se establecen con el comando SCREEN, todos los sprites presentes simultáneamente en la pantalla tienen que tener la misma dimensión. Para obtener sprites de dimensiones superiores a las usadas en los demás, hay que definir más sprites, representando cada uno una parte de la figura, y moverlos juntos por la pantalla.

Midiendo el paso del tiempo

En muchos programas es útil disponer de algún método para medir el tiempo. Por ejemplo, en un juego se puede penalizar al jugador que pierde demasiado tiempo, o, en un programa educativo, hacer aparecer las instrucciones después de que haya pasado un período de tiempo dado desde la última vez que se ha pulsado una tecla, etc.

El método más clásico, y que funciona en todos los dialectos BASIC, es el de usar un bucle FOR...NEXT sin instrucciones. Puede parecer extraño, pero inclusive para no hacer nada el BASIC tarda un tiempo. La explicación de ello está relacionada con la forma en la que el ordenador trata las instrucciones, y está fuera de los objetivos de este libro explicarlo. De todas formas, el extraño efecto se puede utilizar en un programa para obtener un retardo de duración preestablecida, de esta manera:

```
FOR I=1 TO XX:NEXT I
```

donde XX establece el tiempo de espera: cuanto mayor sea el valor de XX, tanto más habrá que esperar para que la ejecución vuelva a empezar.

El BASIC MSX pone a nuestra disposición otros métodos para la medición del tiempo. El primero está representado por la variable de sistema: TIME, que se incrementa en 1 cada vez que el procesador de vídeo genera una interrupción, es decir, cada 1/50 de segundo. Pero ¡cuidado!, en el transcurso de operaciones tales como la lectura o escritura de ficheros desde casete no se generan interrupciones de vídeo, lo que hace que la variable TIME no se incremente.

El último método para medir el tiempo y que resulta muy útil a la hora de emprender una acción determinada es la interrupción ON INTERVAL=XX GOSUB, donde XX es el período deseado expresado en 1/50 de segundo. Esta interrupción, al igual que las demás, requiere una instrucción de activación INTERVAL ON.

Ordenación de los datos

```
320 REM
330 REM
340 REM
350 FOR I=1 TO N-1
360   FOR J=I+1 TO N
370   IF A$(J)<A$(I) THEN SWAP A$(J),A$(I)
380   NEXT J
390 NEXT I
```

Un problema que se presenta prácticamente en todas las aplicaciones de gestión es el de ordenar un conjunto de datos en base a algún criterio preestablecido, alfabético, numérico o reconducible a uno de estos dos tipos.

El programa de ordenación que le proponemos lo puede utilizar en su programa como subrutina. En la versión que les presentamos arriba, ordena alfabéticamente cadenas de caracteres, pero es suficiente cambiar el tipo de la variable para poder ordenar también números.

Indicamos con A el vector, y con N el número de elementos, mientras I y J son dos variables que identifican los dos elementos que se comparan. Suponemos que el vector ya está memorizado (en caso contrario sería suficiente realizar la subrutina de lectura). Se compara el primer elemento del vector con todos los demás; cuando se encuentra uno menor se efectúa el cambio de posición entre los dos, y se siguen realizando las comparaciones pero usando como término de comparación no el primer elemento, sino el que lo ha sustituido. Después de haber efectuado todas las posibles comparaciones, en primera posición estará sin duda el elemento menor de todos. Luego se repite el ciclo de comparaciones partiendo del segundo elemento y, al final, en segunda posición estará el elemento menor de todos los que quedaban. Siguiendo así, los elementos del vector se van disponiendo uno a uno en orden creciente.

¿Música o ruido?

```
10 CLEAR 1000
20 A$="V15O4S1M1000T100L40"
```

```
30 B$="CCFCFAFR25FFAFA"
40 C$="O5CO4AR25FAO5CO4R25"
50 D$="AFCR25CCFR25L30FR35L10F"
60 FOR I=1 TO 1
70 PLAY "XA$;"
80 PLAY "XB$;"
90 PLAY "XC$;"
100 PLAY "XD$;"
110 NEXT I
```

Desde el punto de vista físico, el sonido está representado por meras vibraciones del aire que llegan a nuestro oído, donde se traducen en sensaciones sonoras gracias al aparato auditivo. Su MSX no genera vibraciones del aire directamente, sino señales eléctricas que tienen que llegar al altavoz de la pantalla para que el hombre las pueda percibir. Los sonidos y, por tanto, las señales eléctricas que los generan, se caracterizan por tres parámetros:

- Altura: es la frecuencia del sonido. Un sonido de frecuencia alta se percibe como agudo; uno de frecuencia baja, como grave.
  Las frecuencias que el oído humano puede distinguir van de 70 a 13.000 Hertz normalmente (1 Hertz, abreviado Hz, significa una vibración por segundo). Más allá de los 16.000 Hz se encuentran los ultrasonidos, que sólo son percibidos por algunos animales, como los perros y los murciélagos; por debajo de los 20 Hz se colocan los infrasonidos, producidos, por ejemplo, por las olas del mar y por los terremotos.
- Intensidad: se percibe como volumen del sonido. Cuando se actúa sobre el mando de volumen de una radio se varía, de hecho, la intensidad de las señales eléctricas que llegan al altavoz.
- Timbre: es la estructura del sonido. El timbre permite distinguir dos instrumentos distintos que producen la misma nota. El La de un violín es, sin lugar a duda, muy distinto del La de un piano, no sólo por una cuestión de frecuencias, sino también de timbre.

Estas tres magnitudes se pueden variar fácilmente utilizando, en la instrucción PLAY, el subcomando correspondiente, representado generalmente por una letra y un número.

Las posibilidades musicales del MSX son muchas y superiores a las de la mayoría de los ordenadores que se encuentran en el mercado. El secreto de estas prestaciones excelentes reside en la utilización de un microprocesador dedicado única y exclusivamente a la generación de sonidos.

Un aspecto muy atractivo de los MSX es, además, la posibilidad de comunicar con dicho dispositivo de forma bastante natural (por lo menos para el que sabe un poco de música) y no a base de difíciles y aburridas PEEK y POKE a oscuras locaciones de memoria. El BASIC MSX dispone, en efecto, de una instrucción PLAY que, en realidad, es algo más que una simple instrucción: es un macrolenguaje, es un sistema para comunicarse con el microprocesador en forma humana. Por ejemplo, para tocar una nota es suficiente escribir el nombre de la nota (en su notación anglosajona, eso sí). Recordemos la correspondencia con las notas europeas:

DO	RE	MI	FA	SOL	LA	SI
C	D	E	F	G	A	B

La estructura del programa que les presentamos es muy sencilla. Primero hay que reservar un espacio adecuado en memoria para las cadenas que definen la melodía con la instrucción CLEAR. Recuerde que el BASIC MSX reserva inicialmente sólo 200 bytes (es decir, 200 caracteres) para las cadenas, y que si excede la capacidad de memoria, obtendrá un mensaje de error. Con CLEAR 1.000 reservamos, en cambio, un espacio de 1.000 caracteres. A continuación se definen las cadenas que contienen los subcomandos musicales, que veremos con más detalle después; siguen las instrucciones PLAY para las distintas cadenas, ubicadas en un ciclo FOR...NEXT, lo que permite, si lo deseamos, repetir más veces el mismo pasaje musical. Especificando como valores inicial y final el mismo número, la melodía se ejecutará una sola vez.

En la primera cadena, A$, se definen los parámetros iniciales:

- V15 volumen máximo (controla la intensidad),
- O5 quinta octava (controla la frecuencia),
- S13 envolvente nº 13 de la señal eléctrica (da el timbre al sonido),
- T37 número de notas por unidad de tiempo (da el ritmo a la música),
- L20 establece la duración de la nota (contribuye al timbre).

En las cadenas siguientes, de B a D, se encuentran las notas que se tienen que tocar, con las especificaciones de duración (subcomando L) y ritmo (subcomando T), además de las pausas que hay que efectuar entre un grupo de notas y otro, que también contribuyen al ritmo.

APENDICE A

CODIGOS ASCII

CODIGO Nº	CARACTER	CODIGO Nº	CARACTER
32		45	-
33	!	46	.
34	"	47	/
35	#	48	0
36	$	49	1
37	%	50	2
38	&	51	3
39	'	52	4
40	(	53	5
41	)	54	6
42	*	55	7
43	+	56	8
44	,	57	9

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
58	:	79	O
59	;	80	P
60	<	81	Q
61	=	82	R
62	>	83	S
63	?	84	T
64	@	85	U
65	A	86	V
66	B	87	W
67	C	88	X
68	D	89	Y
69	E	90	Z
70	F	91	[
71	G	92	\
72	H	93	]
73	I	94	^
74	J	95	_
75	K	96	`
76	L	97	a
77	M	98	b
78	N	99	c

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
100	d	121	y
101	e	122	z
102	f	123	{
103	g	124	¦
104	h	125	}
105	i	126	~
106	j	127	⌂
107	k	128	Ç
108	l	129	ü
109	m	130	é
110	n	131	â
111	o	132	ä
112	p	133	à
113	q	134	å
114	r	135	ç
115	s	136	ê
116	t	137	ë
117	u	138	è
118	v	139	ï
119	w	140	î
120	x	141	ì

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
142	Ä	163	ú
143	Å	164	ñ
144	É	165	Ñ
145	æ	166	ª
146	Æ	167	º
147	ô	168	¿
148	ö	169	⌐
149	ò	170	¬
150	û	171	½
151	ù	172	¼
152	ÿ	173	¡
153	Ö	174	«
154	Ü	175	»
155	¢	176	Ã
156	£	177	ã
157	¥	178	Ĩ
158	₧	179	ĩ
159	ƒ	180	õ
160	á	181	ö
161	í	182	Ũ
162	ó	183	ũ

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
184	Ĳ	205	▼
185	ĳ	206	▲
186	¾	207	▶
187	∽	208	◀
188	◊	209	⧗
189	‰	210	⧓
190	¶	211	▪
191	§	212	▪
192	▂	213	▪
193	▚	214	▪
194	▄	215	✲
195	▬	216	Δ
196	▪	218	ω
197	▄	219	█
198	▏	220	▄
199	▞	221	▌
200	█	222	▐
201	▕	223	▀
202	█	224	α
203	▨	225	β
204	▧	226	Γ

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
227	π	241	±
228	Σ	242	≥
229	σ	243	≤
230	µ	244	⌠
231	τ	245	⌡
232	Φ	246	÷
233	θ	247	≈
234	Ω	248	°
235	δ	249	•
236	∞	251	√
237	ø	252	ⁿ
238	∈	253	²
239	∩	254	■
240	≡	255	

IMPRESORA SONY

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
32		53	5
33	!	54	6
34	"	55	7
35	#	56	8
36	$	57	9
37	%	58	:
38	&	59	;
39	'	60	<
40	(	61	=
41	)	62	>
42	*	63	?
43	+	64	@
44	,	65	A
45	-	66	B
46	.	67	C
47	/	68	D
48	0	69	E
49	1	70	F
50	2	71	G
51	3	72	H
52	4	73	I

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
74	J	96	`
75	K	97	a
76	L	98	b
77	M	99	c
78	N	100	d
79	O	101	e
80	P	102	f
81	Q	103	g
82	R	104	h
83	S	105	i
84	T	106	j
85	U	107	k
86	V	108	l
87	W	109	m
88	X	110	n
89	Y	111	o
90	Z	112	p
91	[	113	q
92	¥	114	r
93	]	115	s
94	^	116	t
95	_	117	u

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
118	v	140	ゃ
119	w	141	ゅ
120	x	142	ょ
121	y	143	っ
122	z	144	
123	{	145	あ
124	¦	146	い
125	}	147	う
126	~	148	え
127		149	お
128	♠	150	か
129	♥	151	き
130	♣	152	く
131	♦	153	け
132	○	154	こ
133	●	155	さ
134	を	156	し
135	ぁ	157	す
136	ぃ	158	せ
137	ぅ	159	そ
138	ぇ	160	
139	ぉ	161	｡

CODIGO Nº	CARACTER	CODIGO N.º	CARACTER	CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
162	｢	184	ｸ	206	ﾎ	228	と
163	｣	185	ｹ	207	ﾏ	229	な
164	､	186	ｺ	208	ﾐ	230	に
165	･	187	ｻ	209	ﾑ	231	ぬ
166	ｦ	188	ｼ	210	ﾒ	232	ね
167	ｧ	189	ｽ	211	ﾓ	233	の
168	ｨ	190	ｾ	212	ﾔ	234	は
169	ｩ	191	ｿ	213	ﾕ	235	ひ
170	ｪ	192	ﾀ	214	ﾖ	236	ふ
171	ｫ	193	ﾁ	215	ﾗ	237	へ
172	ｬ	194	ﾂ	216	ﾘ	238	ほ
173	ｭ	195	ﾃ	217	ﾙ	239	ま
174	ｮ	196	ﾄ	218	ﾚ	240	み
175	ｯ	197	ﾅ	219	ﾛ	241	む
176	ｰ	198	ﾆ	220	ﾜ	242	め
177	ｱ	199	ﾇ	221	ﾝ	243	も
178	ｲ	200	ﾈ	222	ﾞ	244	や
179	ｳ	201	ﾉ	223	ﾟ	245	ゆ
180	ｴ	202	ﾊ	224	た	246	よ
181	ｵ	203	ﾋ	225	ち	247	ら
182	ｶ	204	ﾌ	226	つ	248	り
183	ｷ	205	ﾍ	227	て	249	る

CODIGO N.º	CARACTER	CODIGO N.º	CARACTER
250	れ	253	
251	ろ	254	
252	わ	255	



BIBLIOGRAFIA

MSX programación básica.
J. Pearce y G. Bland. *Paraninfo.*

Lenguaje Máquina MSX. Introducción y conceptos avanzados.
J. Pritchard. *Anaya Multimedia.*

MSX: guía del programador y manual de referencia.
Sato, Murrel. *Anaya Multimedia.*

Descubre tu MSX. Programación y aplicaciones.
J. Pritchard. *Anaya multimedia.*

El libro gigante de los juegos MSX.
Lacey. *Anaya Multimedia.*

El BASIC de la A a la Z.
Boisgontrer. *Elisa.*

18 juegos dinámicos para tu MSX.
Monsaut. *Moray.*

Diccionario de informática.
Masson.

Diccionario de informática Inglés-Español-Francés.
G. A. Nansa. *Paraninfo.*

Diccionario del BASIC.
David A. Lren. *Elisa.*

Diccionario del BASIC.
W. Hart. *Paraninfo.*

INDICE GENERAL

10 **Sistemas operativos y software de base**
Qué son, para qué sirven. Unos desconocidos muy importantes.
11 **Sistema operativo CP/M**
Uno de los sistemas operativos para microprocesadores de 8 bits de mayor difusión en el mercado.
12 **MS-DOS: el estándar de IBM**
Sistema operativo para el microprocesador de 16 bits 8088, adoptado por el IBM-PC.
13 **Paquetes de aplicaciones. Software "pret a porter"**
Características y peculiaridades de los más importantes paquetes de aplicaciones.
14 **VisiCalc: una buena hoja de cálculo**
Interioridades y manejo de una de las hojas de cálculo más usadas.
15 **Dibujar con el ordenador**
Profundizando en una de las facetas útiles y divertidas que nos ofrecen los ordenadores.
16 **Tratamiento de textos... para escribir con el ordenador**
Cómo convertir su ordenador en una máquina de escribir con memoria y todo tipo de posibilidades.
17 **Diseño de juegos**
Particularidades características de esta aplicación de los ordenadores.
18 **LOGO: la tortuga inteligente**
Un lenguaje conocido por su «cursor gráfico», la tortuga, y sus aplicaciones pedagógicas al alcance de su mano.
19 **BASIC y tratamiento de imágenes**
Todo lo que en **¡Y llegó el BASIC!** no se pudo ver sobre las imágenes y gráficos en el BASIC.
20 **Bancos de datos (I)**
Peculiaridades de una de las aplicaciones de los ordenadores más interesantes, y que más dinero mueven.
21 **Bancos de datos (II)**
Profundizando en sus características.
22 **Paquetes integrados: Lotus 1-2-3 y Simphony**
Estudio de dos de los paquetes integrados (Hoja de cálculo + base de datos + ...) más conocidos.
23 **dBASE II y dBASE III**
Cómo aprovechar las dos versiones más recientes de esta importante base de datos.
24 **Los ordenadores uno a uno**
Un amplio y completo estudio comparativo.
25 **Cálculo numérico en BASIC**
Una aplicación especializada a su disposición.

26 **Multiplan**
Cómo hacer uso de este moderno paquete de aplicaciones.
27 **FORTRAN y COBOL**
Dos lenguajes muy especializados y distintos.
28 **FORTH: anatomía de un lenguaje inteligente**
Principales características de un lenguaje moderno, flexible y de amplio uso, en la robótica.
29 **Cómo realizar nuestro propio banco de datos**
Conocimientos necesarios para poder fabricar un banco de datos a nuestro gusto y medida.
30 **Los paquetes integrados uno a uno**
Todos los que usted puede encontrar en el mercado.

NOTA: Ingelek, S. A. se reserva el derecho de modificar, sin previo aviso, el orden, titulo o contenido de cualquier volumen de la colección.

La demanda de equipos y programas intercambiables se ha acentuado enormemente en los últimos años. Los usuarios desean no verse constreñidos únicamente al hardware o software que realice el fabricante del ordenador que compraron.

Fruto de esta situación ha sido la aparición de un estándar para equipos del tipo "doméstico": el MSX. Las especificaciones que determina cubren tanto el soporte hardware (microprocesador, control de gráficos, de sonido, periféricos, etc.) como el lenguaje de programación (BASIC MSX de Microsoft).

Sus evidentes ventajas y el respaldo de marcas tan conocidas como Sony, Toshiba, Philips, etc. han sido, sin duda, algunas de las causas de su aceptación. Día a día aumentan los fabricantes que se acogen a esta norma que, en sus características más importantes, explicamos en el presente volumen de la Biblioteca Básica Informática.

375 pts.